Examples for the SIMATIC S7-1200 / S7-1500 Web Server

STEP 7 Basic (TIA Portal), STEP 7 Professional (TIA Portal)

https://support.industry.siemens.com/cs/ww/en/view/68011496

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

© Siemens AG 2018 All rights reserved

Table of Contents

Legal information **2**

1 Preface **5**

2 Hardware and Software Components Used **6**

3 Basics of Standard Web Pages **8**

4 Basics of User-Defined Web Pages **9**

4.1 Login and instructions for use 9
4.2 Web server – initializing the WWW system function 9
4.3 Libraries – jQuery and S7 Framework 10

5 Reading and Writing with different Variable Types **12**

5.1 Automation task 12
5.2 Functional mechanisms and use 13
5.2.1 S7 program structure 15
5.2.2 User-defined web page (HTML file) structure 16

6 Displaying the Date and Time **19**

6.1 Automation task 19
6.2 Functional mechanisms and use 19
6.2.1 S7 program structure 20
6.2.2 User-defined web page (HTML file) structure 21

7 Displaying an S7 Array (S7-1500 only) **22**

7.1 Automation task 22
7.2 Functional mechanisms and use 22
7.2.1 S7 program structure 23
7.2.2 User-defined web page (HTML file) structure 23

8 ENUM – Variable Value Replaced with Text **24**

8.1 Automation task 24
8.2 Functional mechanisms and use 25
8.2.1 S7 program structure 26
8.2.2 User-defined web page (HTML file) structure 26

9 HTTP Redirection following an Error (S7-1500 only) **27**

9.1 Automation task 27
9.2 Functional mechanisms and use 28
9.2.1 S7 program structure 29
9.2.2 User-defined web page (HTML file) structure 30

10 Change Language on User-Defined Pages **31**

10.1 Automation task 31
10.2 Functional mechanisms and use 31
10.2.1 S7 program structure 32
10.2.2 User-defined web page structure 32
10.3 Extending the example 37

11 Transferring Data without Reloading Pages Using AJAX **38**

11.1 Automation task 39
11.2 Functional mechanisms and use 40
11.2.1 S7 program structure 41
11.2.2 User-defined web page structure 42

© Siemens AG 2018 All rights reserved

12 Displaying Data of a Datalog as a Graph 47

12.1 Automation task 47
12.2 Functional mechanisms and use 48
12.2.1 S7 program structure 49
12.2.2 User-defined web page structure 50

13 Display Elements Using Scalable Vector Graphics 54

13.1 Programming task 54
13.2 Functional mechanisms and use 54
13.2.1 Sample SVG element – rotary motion of a motor 55

14 S7 Diagnostics and Loading Indicator 58

14.1 Automation task 58
14.2 Functional mechanisms and use 58
14.2.1 S7 program structure 59
14.2.2 User-defined web page structure 59

15 Changing Pictures Using JavaScript 64

15.1 Programming task 64
15.2 Functional mechanisms and use 64

16 Button for CPU Restart 66

16.1 Automation task 66
16.2 Functional mechanisms and use 66
16.2.1 S7 program structure 67
16.2.2 User-defined web page structure 67

17 Login on User-Defined Pages 69

17.1 Automation task 69
17.2 Functional mechanisms and use 69
17.2.1 S7 program structure 70
17.2.2 User-defined page structure 70

18 High-Performance Communication via a String 73

18.1 Automation task 73
18.2 Functional mechanisms and use 74
18.2.1 S7 program structure 75
18.2.2 User-defined web page structure 78

19 Installation 82

19.1 Installing the hardware and software 82
19.2 Installing the application example 83

20 Internet links 84

21 History 85

© Siemens AG 2018 All rights reserved

1 Preface

Objective of the application examples

The application examples in this document show you how to extend your own web pages on the S7-1200/1500 web server.

This document describes the examples provided for downloading. It is recommended to run the examples on a computer and simultaneously analyze them with the aid of this document.

Main contents of the application examples

The application examples cover the following key points:

- Reading and writing with different variable types
- Displaying the time
- Outputting arrays (S7-1500 only)
- Using the ENUM data type
- HTTP redirection following an error (S7-1500 only)
- Change language on user-defined pages
- Transferring data without reloading pages
- Displaying a datalog generated by the controller as a graph
- Creating display elements
- Displaying diagnostic information
- Changing pictures using JavaScript
- Restarting the CPU using a button
- Logging in on the user-defined page
- High-performance data transfer

© Siemens AG 2018 All rights reserved

Advantages

Integrated web server in the S7-1200 and S7-1500

The standard web pages for easy display of service and diagnostic information are enabled with a single click.

In addition, you can create custom, user-defined web pages that are referred to below as “user-defined pages”.

Access from anywhere

A web browser allows you to access the S7 web pages from anywhere in the world.

Application example

Universal use of the application example for the SIMATIC S7-1200 and S7-1500.

Benefits

No additional hardware and software required. The web server can be accessed over large distances using mobile communications devices such as tablet computers, smartphones, etc.

Note The application examples in conjunction with the web server should not and cannot replace an HMI system.

2 Hardware and Software Components Used

The application examples were created with the following components.

Hardware components

Table 2-1: Hardware components used for the examples

Component	No.	Order no.	Note
CPU 1511-1 PN	1	6ES7511-1AK01-0AB0	Firmware V2.0.5
CPU 1214C DC/DC/DC	1	6ES7214-1AG40-0XB0	Firmware V4.2
PG/PC with Ethernet interface	1	-	-
IE FC TP STANDARD CABLE	1	6XV1840-2AH10	IE connecting cable, minimum order quantity: 20m
RJ45 plug connector	2	6GK1901-1BB10-2AA0	Can be moulded

Note For these application examples, you need the current CPU firmware version. Depending on the CPU type, the following entries provide related links to the appropriate downloads:

- S7-1500: https://support.industry.siemens.com/cs/ww/en/view/78301349
- S7-1200: https://support.industry.siemens.com/cs/ww/en/view/107539750

© Siemens AG 2018 All rights reserved

Software components

Table 2-2: Software components used for the examples

Component	No.	Order no.	Note
SIMATIC STEP 7 Professional V14	1	6ES7822-1AA04-0YE5	V14.0 (Update 2)
Software tool for creating HTML files such as FrontPage, Notepad++, …	1	-	Web pages created with Notepad++
Web browser such as Internet Explorer, Mozilla Firefox[1)]	1	-	Created and tested with IE11 and Firefox

[1)] The following web browsers were tested for communication with the CPU:

- Internet Explorer (version 11)
- Mozilla Firefox (version 50)

Note The application examples are optimized for Firefox and IE11.
Using other browsers may require changes regarding their display in the browser.

Overall structure

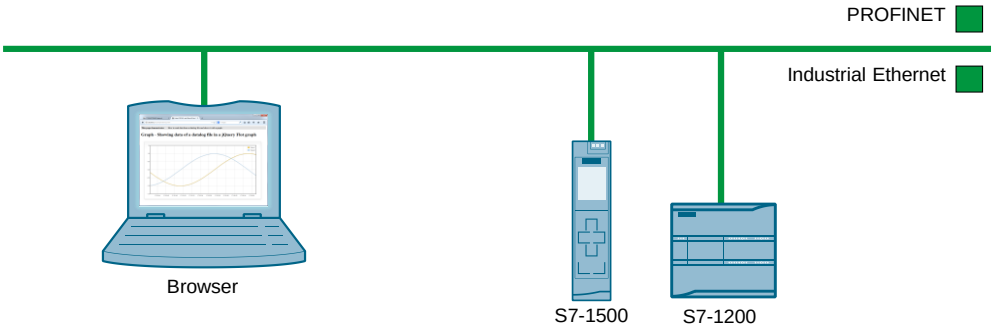
The individual application examples consist of an S7 program running on an S7-1200/1500 and a web page running on the integrated web server of the S7.

Exceptions are the examples in Chapter 13 and 15 that show general web functions without an S7 program.

The web pages are opened using a web browser and consist of an HTML file as a basis.

More complex examples include additional JavaScript files with the “js” file extension.

Figure 2-1: Hardware configuration for the application examples



Sample files and projects

The following list contains the files required for this example, consisting of the S7 project and this PDF document.

Table 2-3: Examples – source files

Component	Comments
68011496_examples_for_S7WebServer_CODE_v21.zip	The file contains the STEP 7 project with the appropriate HTML files in the \html directory.
68011496_examples_for_S7WebServer_en_DOC_v21.pdf	This document.

© Siemens AG 2018 All rights reserved

3 Basics of Standard Web Pages

Requirements

In STEP 7, the following settings are required in the CPU properties.

- The web server must be activated.
- If you require secure access to the web pages, check the “Permit access only with HTTPS” check box.
- Automatic refresh of web pages is enabled.
  The default refresh interval is 10 seconds.
  The interval can be set in the range from 1 to 999s.

Access via HTTP or HTTPS

The URL “http://ww.xx.yy.zz” or “https://ww.xx.yy.zz” provides you with access to the standard web pages. “ww.xx.yy.zz” corresponds to the CPU’s IP address.

HTTPS is used for encrypting and authenticating communication between the browser and the web server. When the “Permit access only with HTTPS” check box is checked, the CPU’s web pages can only be opened via HTTPS.

CPU certificate

If the browser reports a certificate error, proceed as described in the following FAQ:

https://support.industry.siemens.com/cs/ww/en/view/63314183

Access rights for login to the web pages in STEP 7

By factory default, the “Everybody” user is set in each SIMATIC S7-1200/1500 controller.

This user has limited access rights and no password.
(This prevents access to user-defined pages.)

To have full access to user-defined pages, you have to log in with a user who has the required access rights. You can parameterize users, passwords and access rights with STEP 7 in the S7-1200/1500 controller properties.

The login input fields are located in the top left corner of each standard web page of the S7-1200/1500 controller.

Figure 3-1: Login window on standard web page

SIMATIC S7-1200 / S7-1500 standard web pages

Via the integrated standard web page, the S7-1200 and S7-1500 web server already offers a large amount of information about the respective CPU.

The structure of the standard web pages is described in detail in the S7-1500 Web Server Function Manual; it is not the subject of this document.

© Siemens AG 2018 All rights reserved

4 Basics of User-Defined Web Pages

For the basics of user-defined pages, please refer to this application example:
“Creating and using user-defined web pages on S7-1200 / S7-1500”

https://support.industry.siemens.com/cs/ww/en/view/68011496

4.1 Login and instructions for use

In TIA Portal, each example name starts with the chapter number:
Example: Chapter 5 corresponds to program name “05ReadAndWriteS7...”

The web contents for the examples can be found in the appropriate S7 project on your computer in the “…\html_...” folder.

Login of the user-defined pages in the application examples

User: **admin**
Password: **s7**

4.2 Web server – initializing the WWW system function

In the application examples, the “WWW” system function (SFC 99) is called by OB 1.

This block call is described in all examples, including the reference to this chapter.

No.	Function
1	The “WWW” system function (SFC 99) is called in OB1. This function initializes the CPU web server. Cyclically calling the function in OB1 allows interaction between the web server and the controller. The Web DB (DB 333) is connected to the WWW function. The Web DB and the Fragment DB(s) store the structure of the user-defined pages.
2	More instructions and block calls are described in the respective application examples.

Note For basic information about generating data blocks for user-defined pages, please refer to the “First steps” of the following entry:

https://support.industry.siemens.com/cs/ww/en/view/68011496

© Siemens AG 2018 All rights reserved

4.3 Libraries – jQuery and S7 Framework

The web pages of the application examples consist of HTML documents, JavaScript files and JavaScript libraries.

The file structures can be found in the respective application examples.

Some application examples use the “jQuery” and S7 Framework JavaScript libraries. These examples are listed in the following table.

Table 4-1: Libraries used in the application examples

Chapter no.	Chapter name	jQuery	S7 Framework
10	Change Language on User-Defined Page	X	-
12	Displaying Data of a Datalog as a Graph	X	X
14	S7 Diagnostics and Loading Indicator	X	X
16	Button for CPU Restart	X	X
17	Login on User-Defined Page	X	X
18	High-Performance Communication via a String	X	X

Integrate the jQuery library and the appropriate JavaScript files that access or use this jQuery library into your HTML base document. For example, the “S7 Framework” JavaScript file uses the jQuery library; in the HTML code, it is therefore below the call of the jQuery library.

Figure 4-1: Call of JavaScript libraries in the HTML document

```
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <script type="text/javascript" src="script/jquery-2.1.3.min.js"></script>
    <script type="text/javascript" src="script/S7_framework_0.1.7.js"></script>
</head>
```

jquery-2.1.3.min.js (jQuery library)

The “jquery-2.1.3.min.js” file is a library that enables more efficient programming with JavaScript. The library contains basic DOM, event, effects and Ajax functions that are not described in detail in this document. The library’s methods are typically accessed by the prefixed “jQuery” object identifier.
This object identifier is replaced with a “$” sign, which reduces the total number of characters in a JavaScript file.

Example:

jQuery.*post(URL, DATA)*

$.*post(URL, DATA)*

For general information about jQuery, visit: https://jquery.com/

© Siemens AG 2018 All rights reserved

S7_framework.js

The JavaScript file contains a number of functions for converting and processing data from a SIMATIC S7-1200/1500.

The S7 Framework programming uses elements from the jQuery library. Therefore, in the HTML code, always call the S7 Framework after the jQuery library.

There are separate application examples for the S7-1200 and the S7-1500 controller. Both controllers access the same web files.

However, depending on the CPU, access to the default web server differs in terms of IDs, login and other functions. A controller type check is therefore implemented in the examples.

The following table lists the S7 Framework functions.

Table 4-2: S7 Framework functions

No.	Code description
1	Assignment of variable types within S7 Framework

	`var S7Framework = (function($, undefined){`
	`  // type of variable - 0=Bool, 1=unsigned INT, 2=signed INT, 3=real, 4=LReal, 5=String`
	`  var BOOL = 0, UINT = 1, INT = 2, REAL = 3, LREAL = 4, STRING = 5;`

2	PLC functions (different tasks as a library in the framework) - Functions and variables for - AJAX (data transfer) - JSON (data processing of strings using parser) - CPU type (access to S7-1200/1500 web server) - Logon/logoff (referenced to S7 standard web page, “Portal.mwsl”) - Loading indicator (definition and function for animated loading icon) - Error handling within S7 Framework data processing
	`function Controller () {`

Note In the code, you will find information about the functions.
A detailed description of the functions is not the subject of this guide.

© Siemens AG 2018 All rights reserved

5 Reading and Writing with different Variable Types

5.1 Automation task

The task is to read and rewrite variables of different data types.

Note The DTL data type is only supported by the S7-1500.

Requirements for the automation task

Two HTML pages have to be programmed:

- One HTML page that allows you to read/write variables of different types.
- One HTML page that allows you to read special variables.

© Siemens AG 2018 All rights reserved

5.2 Functional mechanisms and use

Overall structure

The application example consists of an S7 program and two HTML documents that are displayed as web pages in a browser.

User-defined page structure

The following figure shows the user-defined page for displaying simple PLC tags that are read/write accessed.
The top “Simple Variables” access S7 flags, whereas the “Structure and Array” variables access a “DB 1” data block.

Figure 5-1: User-defined page for reading/writing simple variables

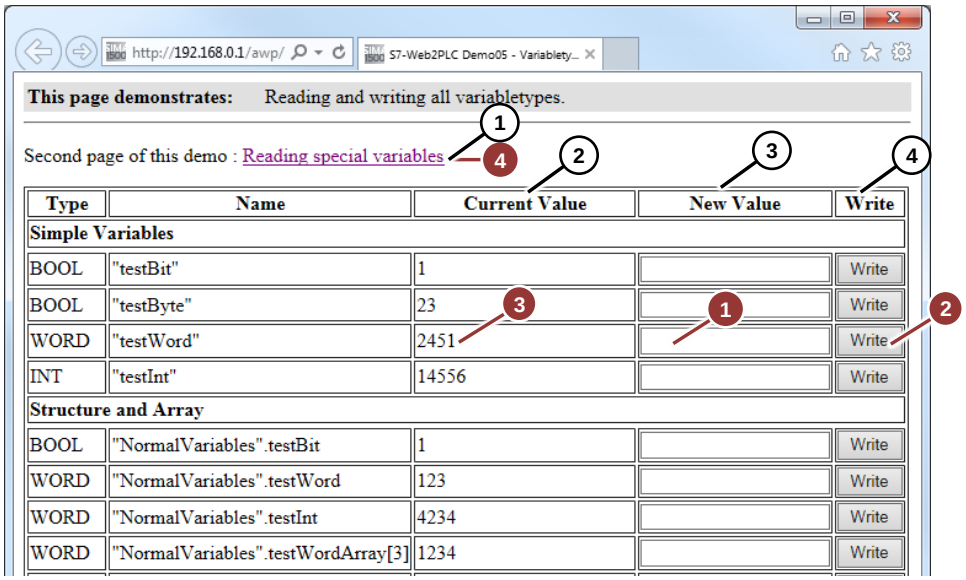

http://192.168.0.1/awp/
S7-Web2PLC Demo05 - Variablety...
This page demonstrates: Reading and writing all variabletypes.
Second page of this demo : Reading special variables
1
2
3
4
Type | Name | Current Value | New Value | Write
Simple Variables
BOOL | "testBit" | 1 | Write
BOOL | "testByte" | 23 | Write
WORD | "testWord" | 2451 | Write
INT | "testInt" | 14556 | Write
Structure and Array
BOOL | "NormalVariables".testBit | 1 | Write
WORD | "NormalVariables".testWord | 123 | Write
WORD | "NormalVariables".testInt | 4234 | Write
WORD | "NormalVariables".testWordArray[3] | 1234 | Write

Table 5-1: Web page for reading/writing simple variables

Item	Description
(1)	This link takes you to the HTML web page with the special variables, “Reading special variables”.
(2)	This column displays the current values of the variables.
(3)	The value to be written is entered in this column.
(4)	The “Write” buttons transfer the values to the CPU.

Table 5-2: Instructions for using the user-defined page

Step	Instruction
1	In the “New Value” column, enter a value corresponding to the data type.
2	Click the “Write” button.
3	The value appears in the “Current Value” column.
4	Click this link if you want to open the “Reading special variables” web page.

© Siemens AG 2018 All rights reserved

The following figure shows the structure of the user-defined page for reading special variables.

Figure 5-2: User-defined page for reading/writing special variables

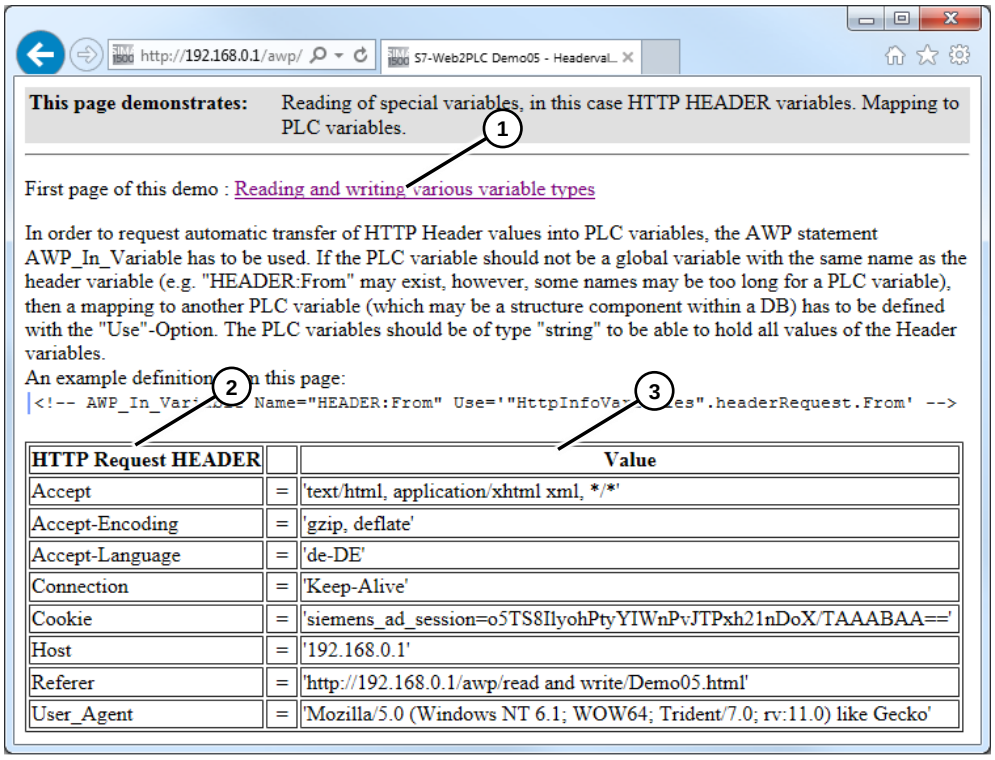

http://192.168.0.1/awp/
S7-Web2PLC Demo05 - Headerval...
This page demonstrates: Reading of special variables, in this case HTTP HEADER variables. Mapping to PLC variables.
1
First page of this demo : Reading and writing various variable types
In order to request automatic transfer of HTTP Header values into PLC variables, the AWP statement AWP_In_Variable has to be used. If the PLC variable should not be a global variable with the same name as the header variable (e.g. "HEADER:From" may exist, however, some names may be too long for a PLC variable), then a mapping to another PLC variable (which may be a structure component within a DB) has to be defined with the "Use"-Option. The PLC variables should be of type "string" to be able to hold all values of the Header variables.
An example definition
this page:
2
3
HTTP Request HEADER
Value
Accept = 'text/html, application/xhtml xml, */*'
Accept-Encoding = 'gzip, deflate'
Accept-Language = 'de-DE'
Connection = 'Keep-Alive'
Cookie = 'siemens_ad_session=o5TS8IlyohPtyYIWnPvJTPxh21nDoX/TAAABAA=='
Host = '192.168.0.1'
Referer = 'http://192.168.0.1/awp/read and write/Demo05.html'
User_Agent = 'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko'

Table 5-3: Web page for reading/writing special variables

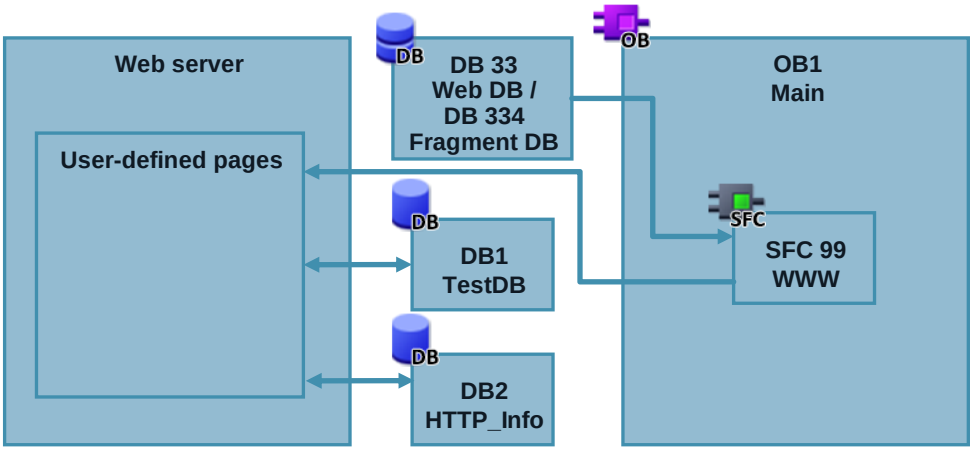
Item	Description
1	This link takes you to the web page for reading/writing PLC tags.
2	This column displays the names of the special variables.
3	This column displays the information (value) of the respective special variable.

© Siemens AG 2018 All rights reserved

5.2.1 S7 program structure

The diagrammatic representation below shows the S7 program structure. The HTML web page is called cyclically. The following sections describe the data exchange with the blocks.

Figure 5-3: S7 program for reading/writing variables



How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

Contents of the PLC tags

The top section of the first HTML page defines simple PLC tags (flags) that can be read/written using the web page.

Contents of DB1

The “TestDB” (DB1) defines variables that can be read/written using the web page. Exception: the “WWW_RET_VAL” variable. This variable contains the return value of the WWW function.

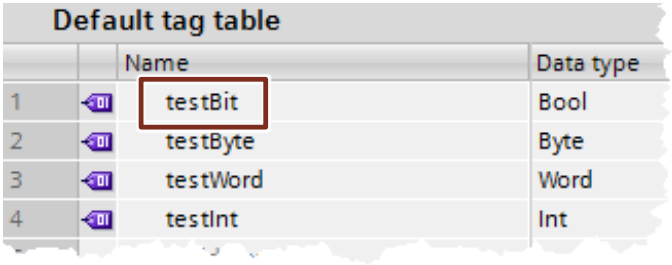
Contents of DB2

DB2 “HTTP_Info” stores the special variables; in this case: HEADER_Request variables.
The user can only read these variables using the web page. These variables are written using the web page via the web server.

© Siemens AG 2018 All rights reserved

5.2.2 User-defined web page (HTML file) structure

No.	Code section

1	To read/write a PLC tag from the PLC tags (tag table) of a web page, only the “variable name” in the syntax is relevant.
	General :=“variable name”:
	
	Example :=“testBit”:
	
	 
	
	`<!-- AWP_In_Variable Name='"testBit"' -->`
	
	`<tr>`
	`<form method="post" action="" >`
	`<td>BOOL</td><td>"testBit"</td><td>:="testBit":</td>`
	`<td><input type="text" name='"testBit"' maxlength="8" ></td>`
	`<td><input type="submit" value="Write"></td>`
	`</form>`
	`</tr>`

© Siemens AG 2018 All rights reserved

No.	Code section
2	To read/write a variable from a DB, the DB name and the variable name are relevant.

NormalVariables

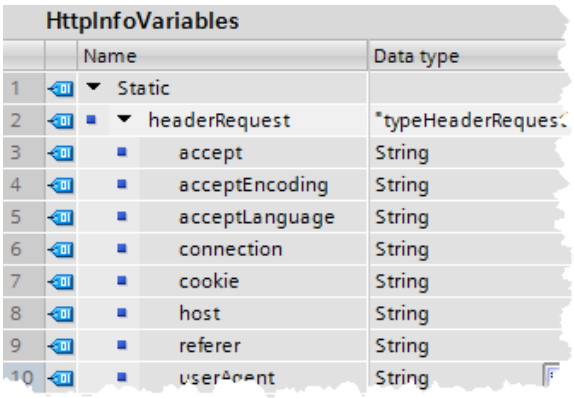
		Name	Data type
1		Static	
2		testBit	Bool
3		testWord	Word
4		testInt	Int
5		testWordArray	Array[0..3] of Word
6		testBitArray	Array[0..3] of Bool
7		testChar	Char
8		testByte	Byte
9		testDWord	DWord
10		testDInt	DInt
11		testReal	Real
12		testString	String
13		testTime	Time
14		testDate	Date
15		testToD	Time_Of_Day
16		testDT	DTL

```
<!-- AWP_In_Variable Name='"NormalVariables".testBit' -->

<tr>
<form method="post" action="">
<td>BOOL</td>
<td>"NormalVariables".testBit</td>
<td>:="NormalVariables".testBit:</td>
<td>
    <input type="text" name='"NormalVariables".testBit' maxlength="8" >
</td>
<td>
    <input type="submit" value="Write">
</td>
</form>
</tr>
```

© Siemens AG 2018 All rights reserved

No.	Code section

3	The special variables (here: HTTP request variables) are saved in the DB variables with UDP commands.
	This DB variable is read by the HTML file.
	
	 
	
	`<!-- AWP_In_Variable Name="HEADER:Accept"`
	`Use='"HttpInfoVariables".headerRequest.accept' -->`
	
	`:="HttpInfoVariables".headerRequest.accept:`
	
	Note:
	http request variables are global variables within a script's scope.

© Siemens AG 2018 All rights reserved

6 Displaying the Date and Time

6.1 Automation task

The task is to display the CPU's time on a web page.

Requirements for the automation task

- Read the time in the STEP 7 program using the "RD_LOC_T" function.
- Program a web page that displays the time.
- Update the time with the aid of a second HTML file (inline frame) integrated in the HTML file.
  For a description of the term 'inline frame', please refer to Chapter 11.

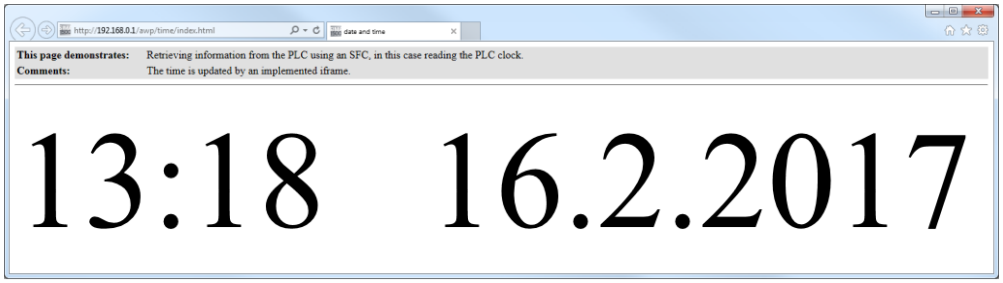
6.2 Functional mechanisms and use

The application example consists of an S7 program and two HTML documents that are displayed as web pages in a browser.

User-defined page structure

The date and time are displayed horizontally and centered on the user-defined page.

Figure 6-1: User-defined page for displaying the date and time


This page demonstrates: Retrieving information from the PLC using an SFC, in this case reading the PLC clock.
Comments: The time is updated by an implemented iframe.
13:18 16.2.2017

Setting the time in the controller

The first step is to set the current time in the data source, i.e., in the CPU.

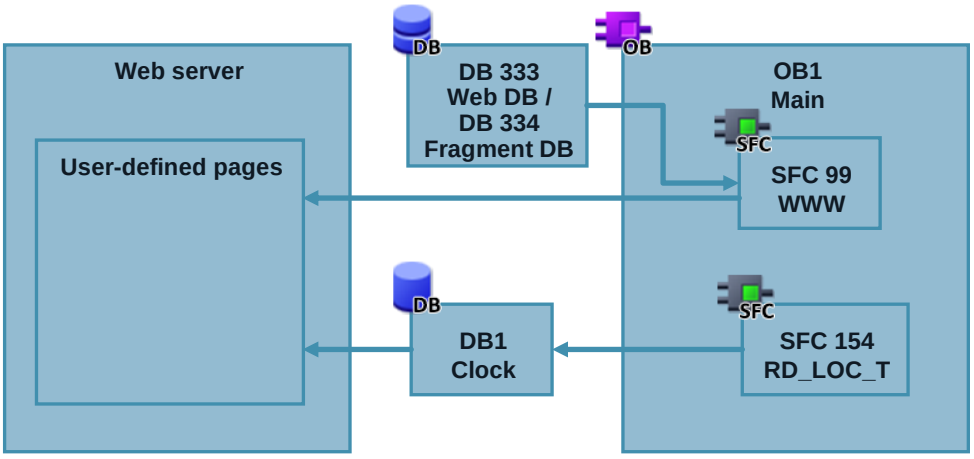
1. Open TIA Portal and the program example.
2. In the Project tree, right-click your controller and select "Go online".
3. Select "Online & diagnostics".
4. In Functions, select "Set time".
5. In Properties, select the time zone and, if necessary, standard time/daylight saving time.

© Siemens AG 2018 All rights reserved

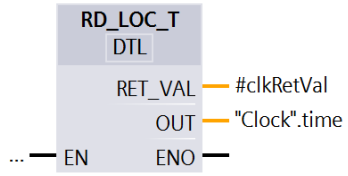
6.2.1 S7 program structure

The diagrammatic representation below shows the S7 program structure. The HTML user-defined page is called cyclically. The following sections describe the data exchange with the blocks.

Figure 6-2: S7 program for displaying the date and time



How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.
2	The “RD_LOC_T” instruction (SFC 154) reads the CPU’s time and saves it to the “Clock” DB (DB 1) to the “time” variable. 

Contents of the Clock DB (DB 1)

This DB stores the time in “DTL” format.
The user-defined page reads the time from this DB.

© Siemens AG 2018 All rights reserved

6.2.2 User-defined web page (HTML file) structure

No.	Code section

1	The UDP commands initialize the variables.
	`<!-- AWP_In_Variable Name='"Clock".time.YEAR' -->`
	`<!-- AWP_In_Variable Name='"Clock".time.MONTH' -->`
	`<!-- AWP_In_Variable Name='"Clock".time.DAY' -->`
	`<!-- AWP_In_Variable Name='"Clock".time.HOUR' -->`
	`<!-- AWP_In_Variable Name='"Clock".time.MINUTE' -->`
	`<!-- AWP_In_Variable Name='"Clock".time.SECOND' -->`

2	The inline frame (“Update_Page.html” file) writes the time to this div box.
	`<div id="clock">`
	`</div>`

3	The inline frame must be integrated into the “index.html” file in order to integrate it into the functional sequence. `<iframe src="Update_Page.html" style="display:none;"/>`

4	The “Update_Page.html” file specifies which variables are cyclically updated and where to store them.
	(This is implemented using the ID in the JavaScript file and in the HTML code.)
	`/* Function for updating the variables */`
	`function updateSingleVariablesTable()`
	`{`
	`var table = document.getElementById("singleVariablesTable");`
	`for(i = 1; i < table.rows.length; i++)`
	
	Referenced to:
	`<table id="singleVariablesTable" border="1">`

	The ‘if’ statement is prefixed for displaying a “0”, e.g. for 12:03h.
	`<td><!-- read of date and time -->`
	`<script type="text/javascript">`
	`var hour = :="Clock".time.HOUR:;`
	`var minute = :="Clock".time.MINUTE:;`
	`if (hour < 10)`
	`{`
	`hour= "0" + hour;`
	`}`
	`if (minute < 10)`
	`{`
	`minute = "0" + minute;`
	`}`
	`document.write(hour + ":");`
	`document.write(minute);`
	`document.write('   :="Clock".time.DAY:.');`
	`document.write(':="Clock".time.MONTH:.');`
	`document.write(':="Clock".time.YEAR:');`
	`</script>`
	`</td>`
	`<td>`
	`clock`
	`</td>`
	
	For information about this update method, please refer to the following FAQ:
	http://support.automation.siemens.com/WW/view/en/97044123

© Siemens AG 2018 All rights reserved

7 Displaying an S7 Array (S7-1500 only)

7.1 Automation task

The task is to read all fields of an S7 array and output them on a user-defined page as a table.

Requirements for the automation task

- Create an array in a DB of the STEP 7 program.
- Display the array fields in a table on a user-defined page.

7.2 Functional mechanisms and use

User-defined page structure

The user-defined page displays the array values structured in a table. The values in the array are updated by refreshing the user-defined page using “F5”.

Figure 7-1: User-defined page for displaying S7 arrays

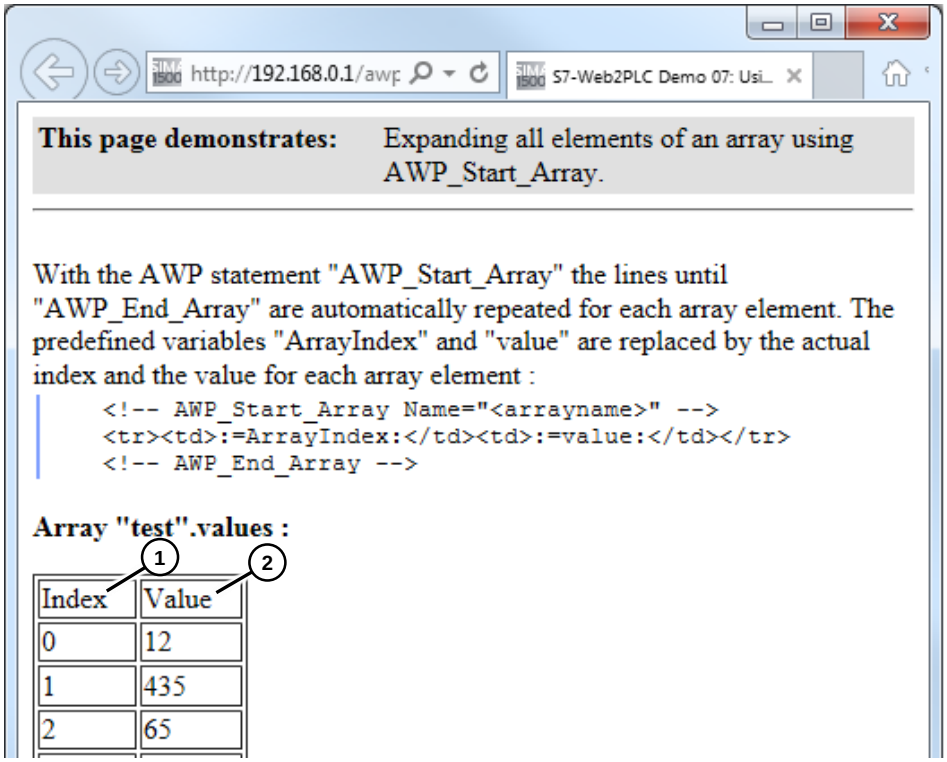


Table 7-1: User-defined page for displaying S7 arrays

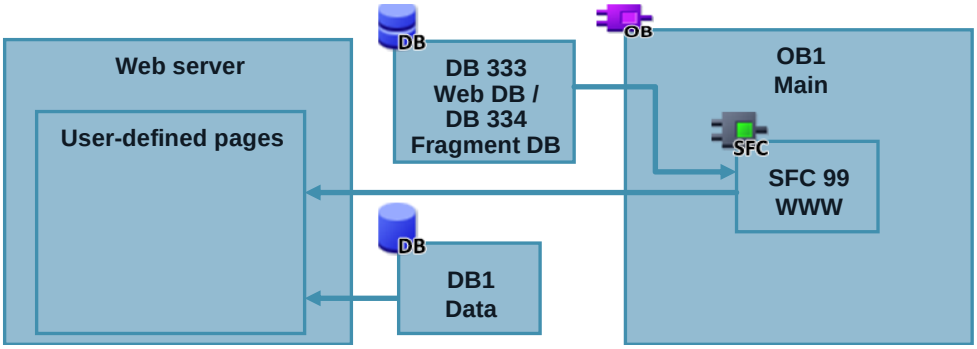
Item	Description
①	This column displays the array index
②	This column displays the array field value

© Siemens AG 2018 All rights reserved

7.2.1 S7 program structure

The diagrammatic representation below shows the S7 program structure. The change of array values is controlled using the DB1 online view and the user-defined page is refreshed as described in Chapter 7.2.

Figure 7-2: S7 program for displaying S7 arrays



How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

Contents of the Data DB (DB 1)

The Data DB1 defines the array that is read by the user-defined page.

7.2.2 User-defined web page (HTML file) structure

The array contains the variables displayed on the user-defined page.

No.	Function
1	Adapt the UDP command to the respective array name (DB). Data Name / Data type / Start value 1 Static 2 values — Array[0..9] of Int 3 values[0] — Int — 12 4 values[1] — Int — 435 5 values[2] — Int — 65 6 values[3] — Int — 67 7 values[4] — Int — 234 8 values[5] — Int — 54 9 values[6] — Int — 214 10 values[7] — Int — 625 11 values[8] — Int — 824 12 values[9] — Int — 633 `<table border="1" width="120px" > <tr><td>Index</td><td>Value</td></tr>  <!-- AWP_Start_Array Name='"Data".values' --> <tr><td>:=ArrayIndex:</td><td>:=value:</td></tr> <!-- AWP_End_Array --> </table>`

© Siemens AG 2018 All rights reserved

8 ENUM – Variable Value Replaced with Text

8.1 Automation task

The task is to change the integer value of a variable using different buttons on a user-defined page.

The values that are written to the variable are to be linked with different message texts.

A text display is to display the appropriate message texts.

Requirements for the automation task

- Read/write a variable defined as ENUM with a user-defined page.
- Define an ENUM variable.
- Create a variable in a PLC tag table.

Note ENUM variables are defined in an HTML file.
In ENUM variables, numerical values are replaced with a string. The S7 program uses a numerical variable.

© Siemens AG 2018 All rights reserved

8.2 Functional mechanisms and use

User-defined page structure

Figure 8-1: ENUM user-defined page – variable values are replaced with text

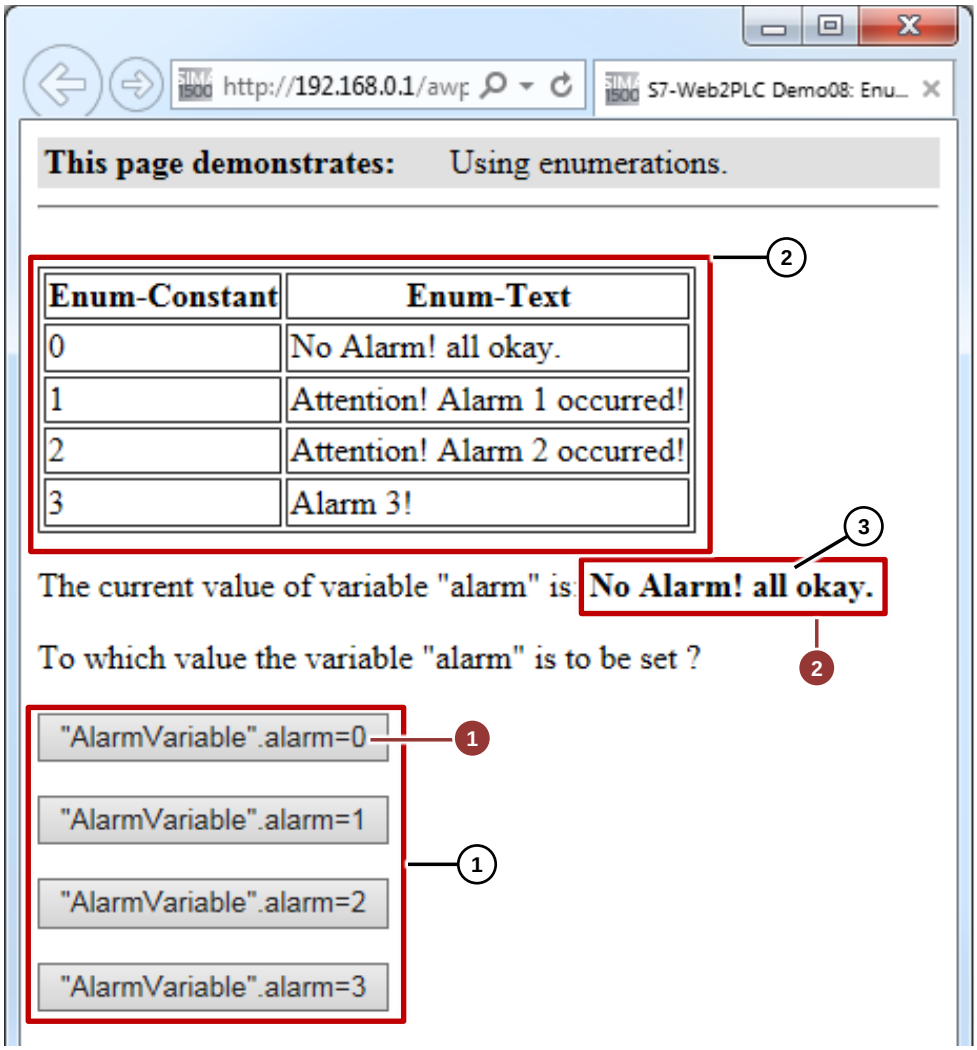


Table 8-1 ENUM user-defined page – variable values are replaced with text

Item	Description
1	These buttons allow you to set the “alarm” variable to a value between 0 and 3.
2	This table shows the assignment of the text to the numerical value.
3	This is where the appropriate text is output.

Table 8-2: Instructions for using the user-defined page

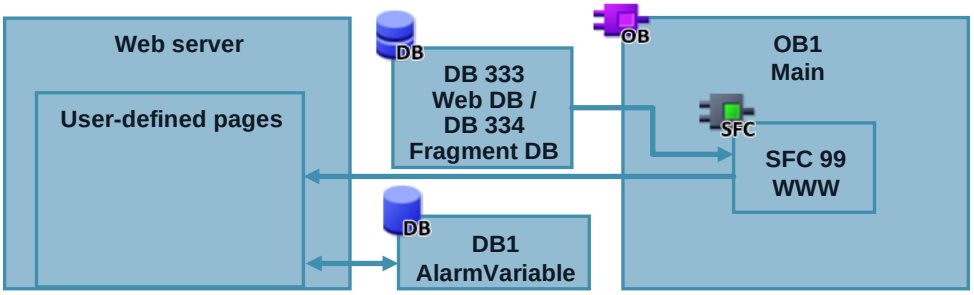
Step	Instruction
1	Select one of the four possible buttons, in this case: “alarm=0”.
2	The “No Alarm! all okay.” text is output based on the numerical assignment, “alarm=0”.

© Siemens AG 2018 All rights reserved

8.2.1 S7 program structure

The diagrammatic representation below shows the S7 program structure. The value of the “alarm” variable in DB1 is set by the field selection of the user-defined page.

Figure 8-2: ENUM S7 program – variable values are replaced with text



How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

Contents of the AlarmVariable DB (DB 1)

No.	Function
1	This DB stores the “alarm” variable that can be read/written by the user-defined page. The variable is modified by the selection of the user-defined page (here: “alarm=0”) AlarmVariable Name | Data type | Start.. | Monitor value Static alarm | Int | 0 | 0

8.2.2 User-defined web page (HTML file) structure

No.	Code section

1	Definition of the ENUM variable in the “_enumdefs.htm” file
	`<!-- AWP_Enum_Def Name="MyAlarmEnum" Values='`
	`0:"No Alarm! all okay.",`
	`1:"Attention! Alarm 1 occurred!",`
	`2:"Attention! Alarm 2 occurred!",`
	`3:"Alarm 3!"' -->`
	`<!-- AWP_Enum_Def Name="abc" Values='0:"null", 1:"one"' -->`

2	Initialization of the ENUM variable `<!-- AWP_In_Variable Name='"AlarmVariable".alarm' Enum="MyAlarmEnum" -->`

3	The value of the PLC tag is read and displayed as text.
	`<p>The current value of variable "alarm" is:`
	`<strong> :="AlarmVariable".alarm:</strong></p>`

4	The assigned text is used when writing the PLC tag. The associated value (digit) is written to the PLC tag.
	`<form method="post" action="">`
	`<p><input type="hidden" name='"AlarmVariable".alarm' value="No Alarm! all okay."></p>`
	`<p><input type="submit" value='"AlarmVariable".alarm=0'><p>`
	`</form>`

© Siemens AG 2018 All rights reserved

9 HTTP Redirection following an Error (S7-1500 only)

9.1 Automation task

If a variable has exceeded a certain value, a new user-defined page is to open displaying an appropriate message.

The value of the variable is to be settable using an input field on the user-defined page.

Requirements for the automation task

- Program a user-defined page with an input field for setting a variable value.
- The ENUM variable is specified by the STEP 7 program.
- An ENUM variable is to control the URL (Uniform Resource Locator).

© Siemens AG 2018 All rights reserved

9.2 Functional mechanisms and use

User-defined page structure

Figure 9-1: User-defined page for HTTP redirection following an error

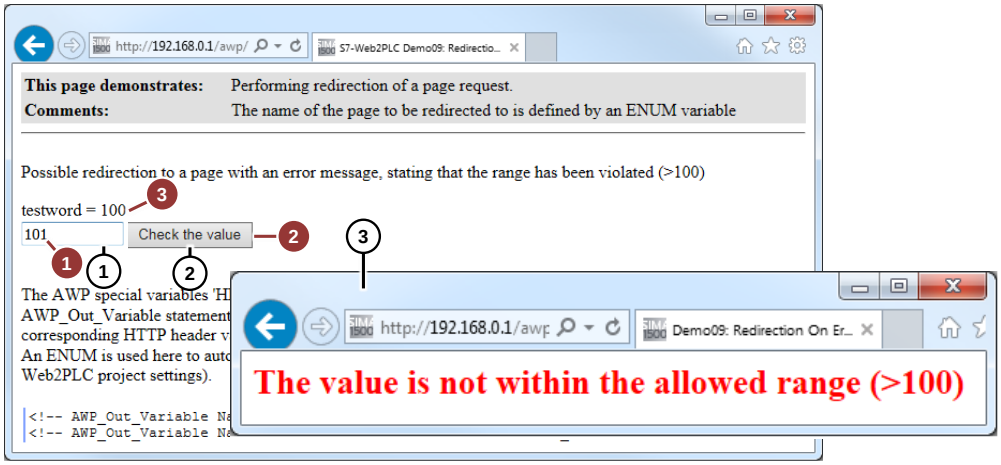

http://192.168.0.1/awp/
S7-Web2PLC Demo09: Redirectio...
This page demonstrates: Performing redirection of a page request.
Comments: The name of the page to be redirected to is defined by an ENUM variable
Possible redirection to a page with an error message, stating that the range has been violated (>100)
testword = 100
101
Check the value
1
2
3
http://192.168.0.1/awp
Demo09: Redirection On Er...
The value is not within the allowed range (>100)

Table 9-1: User-defined page for HTTP redirection following an error

Item	Description
(1)	In this field, enter a new input value.
(2)	The “Check the value” button sends the value to the CPU.
(3)	If the input value is greater than “100”, a new user-defined page opens. In this example, the new user-defined page has red text.

Table 9-2: Instructions for using the user-defined page

Step	Instruction
1	In the input field, first enter a value less than “100”.
2	Select the button and repeat the step with “101”.
3	The new value is displayed and the new user-defined page opens if the input value > “100”.

Note

If the checked variable is influenced by the controller, the user-defined page has to be cyclically refreshed. Otherwise, no HTTP redirect will take place.

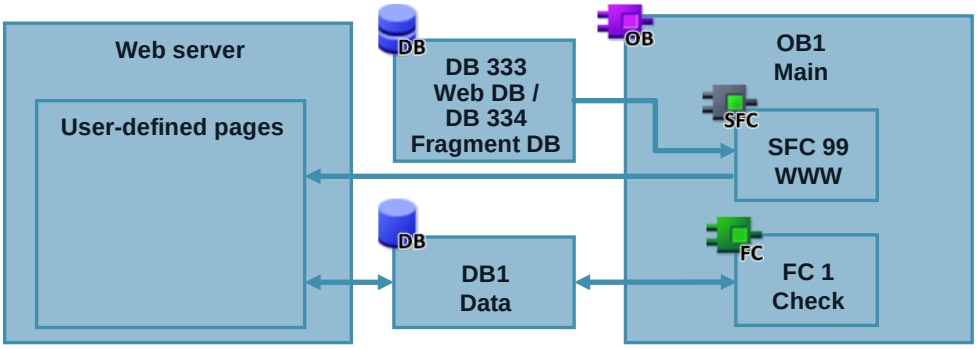
To go to the home page, set the variable value in TIA Portal to less than “100” and reload the user-defined page.

© Siemens AG 2018 All rights reserved

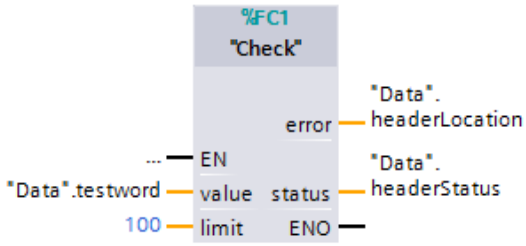
9.2.1 S7 program structure

The diagrammatic representation below shows the S7 program structure. By comparing the values in FC1, the system switches to the user-defined page.

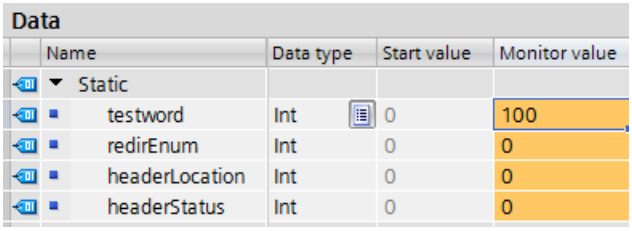
Figure 9-2: S7 program for HTTP redirection following an error



How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.
2	The “Check” function determines whether the “100” limit was exceeded. If this is the case, the “headerLocation” variable is set to “1” and “headerStatus” is set to “302”. Due to this, the system switches to the user-defined page with the error message. If the value was not exceeded, both variables are set to “0”. 

Contents of the Data DB (DB 1)

No.	Function
1	DB1 stores the variables for FC 1. If the value of the “testword” variable > 100, the system switches the user-defined page. 

Data

	Name	Data type	Start value	Monitor value
	Static			
	testword	Int	0	100
	redirEnum	Int	0	0
	headerLocation	Int	0	0
	headerStatus	Int	0	0

© Siemens AG 2018 All rights reserved

9.2.2 User-defined web page (HTML file) structure

No.	Code section

1	To change the location, the appropriate URL must be assigned to the numbers stored in the “headerLocation” variable.
	
	An ENUM variable must be defined for this purpose.
	`<!-- AWP_Enum_Def Name="redirEnum" Values='`
	`1:"Demo09_redir.html",`
	`0:"Demo09.html"' -->`

2	Initialization of the CPU variables and assignment of the ENUM variable.
	`<!-- AWP_In_Variable Name='"Data".testword' -->`
	`<!-- AWP_Out_Variable Name="HEADER:Status"`
	`Use='"Data".headerStatus' -->`
	`<!-- AWP_Out_Variable Name="HEADER:Location"`
	`Use='"Data".headerLocation' Enum="redirEnum" -->`

3	Description of the variable to be checked.
	`<form method="POST" action="">`
	`<input type="text" name='"Data".testword' size="10">`
	`<input type="submit" value="Check the value" >`
	`</form>`

© Siemens AG 2018 All rights reserved

10 Change Language on User-Defined Pages

10.1 Automation task

The task is to implement a change language option (German/English) with flag icons on the user-defined pages and add custom languages to this option.

Requirements for the automation task

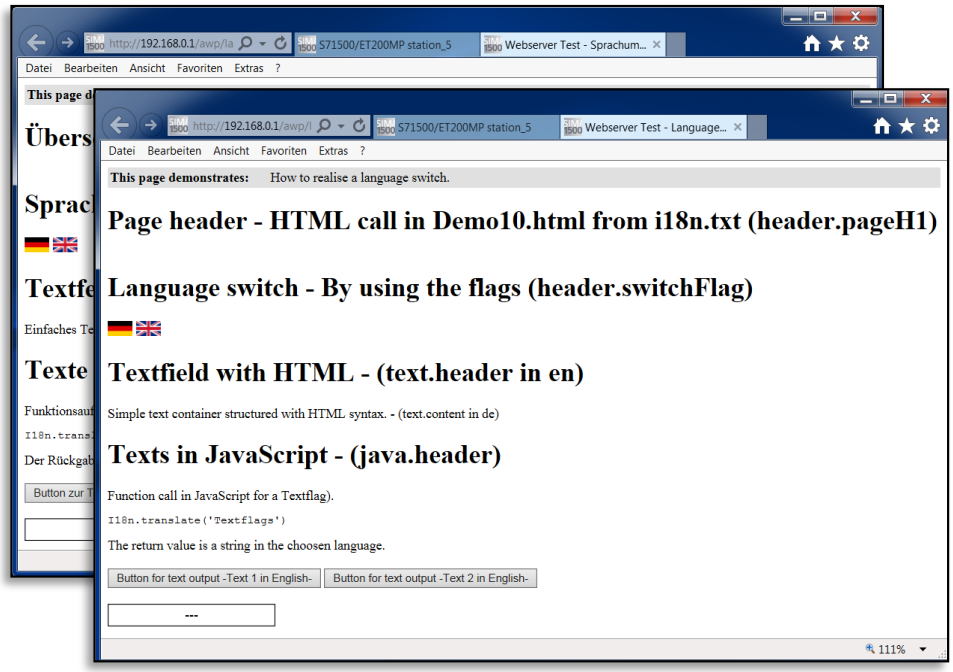
- Program a user-defined page without texts.
- Create a text file with all texts in German and English.
- Change the language using the country's flag.
- Translate texts selected by JavaScript.
- Synchronize the change language option of the user-defined pages with the change language option of the standard web pages.
- Expansion capability of the change language option.

10.2 Functional mechanisms and use

User-defined page structure

On the user-defined page, you can change between German and English. To do this, click the appropriate flag icon.

Figure 10-1: User-defined page for change language


Webserver Test - Language...
Datei Bearbeiten Ansicht Favoriten Extras ?
This page demonstrates: How to realise a language switch.
Page header - HTML call in Demo10.html from i18n.txt (header.pageH1)
Language switch - By using the flags (header.switchFlag)
Textfield with HTML - (text.header in en)
Simple text container structured with HTML syntax. - (text.content in de)
Texts in JavaScript - (java.header)
Function call in JavaScript for a Textflag).
I18n.translate('Textflags')
The return value is a string in the choosen language.
Button for text output -Text 1 in English-
Button for text output -Text 2 in English-

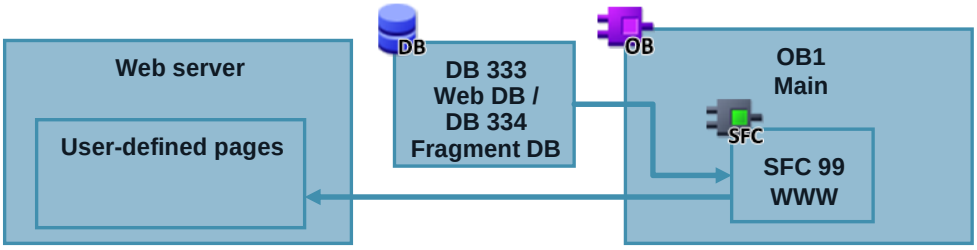
111%

© Siemens AG 2018 All rights reserved

10.2.1 S7 program structure

In the S7 program, only the WWW function is called.

Figure 10-2: S7 program for change language



How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

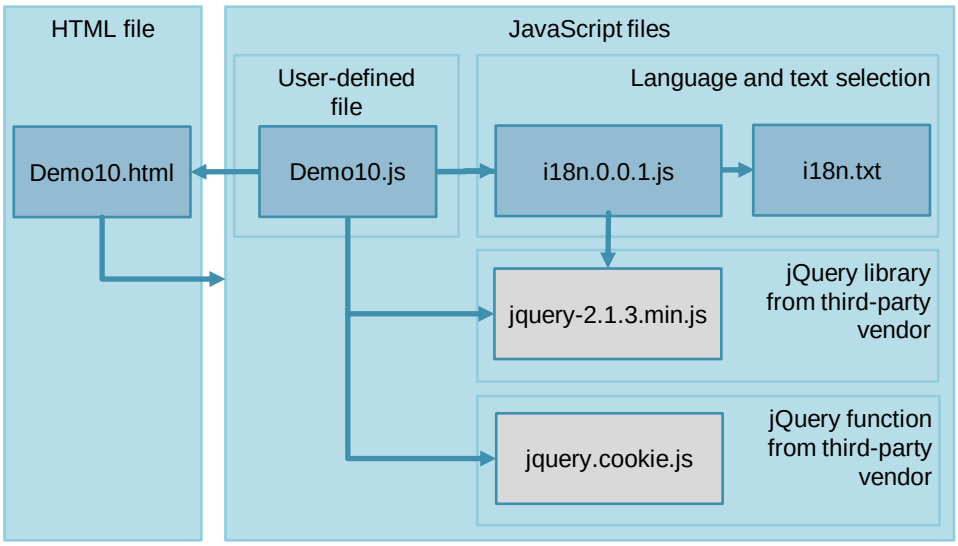
10.2.2 User-defined web page structure

The diagrammatic overview below shows the basic file structure of the user-defined pages.

The term “i18n” that recurs in the following sections stands for a software development process for implementing the change language option independently of the actual program code. The term “i18n” results from internationalization, where 18 stands for the number of letters between “i” and “n”.

For efficient programming of the “Demo10.js” script code and compatible programming of the function of the user-defined pages across browsers, the jQuery library for JavaScript is integrated.

Figure 10-3: Diagrammatic overview of the user-defined web page



© Siemens AG 2018 All rights reserved

Demo10.html

The HTML document defines the structure of the user-defined page and the calls of the JavaScript files used, see Chapter 10.2.2.1.
Throughout the file structure, the language assignment is ensured by the “de” and “en” identifiers.

Demo10.js

Contains the interactive functions used to implement the user inputs, see Chapter 10.2.2.2.
Also defines the behavior of the cookie used. The JavaScript file’s code syntax is supported by the jQuery library.

i18n.txt

The “i18n.txt” text file contains the multilingual text elements, “Demo10.html” and the script functions that are called from “i18n:0.0.1.js”.

i18n.0.0.1.js

Contains the functions for changing the language and transferring the texts with the specific language from “i18n.txt” to the HTML document.

Figure 10-4: Code element for change language

```
// replace text of all elements with data-i18n attribut
function translateLabels() {
    var item, text, key;
    $("[data-i18n]").each(function() {

// Load languagge definition file to var dictionary
function load() {
    var filename = "script/i18n.txt";
    //$.ajax({ type: "POST", url: url, data: "", dataType: "text"})
```

jquery.cookie.js

The JavaScript file provides functions for defining and managing cookies.

In this context, the “Demo10.js” JavaScript file uses these functions and creates or modifies the existing cookie of the standard web page, see Chapter 10.2.2.2. For more information about the JavaScript file used here, visit the following web page:

- https://github.com/carhartl/jquery-cookie
  - https://github.com/carhartl/jquery-cookie/blob/d72bb07e29962330db61aae1eda07f6312419840/src/jquery.cookie.js
  - https://github.com/js-cookie/js-cookie/tree/master/src

jquery-2.1.3.min.js (jQuery library)

The jQuery library file allows more efficient programming with JavaScript. For more information about the jQuery library, please refer to Chapter 4.3.

© Siemens AG 2018 All rights reserved

10.2.2.1 Demo10.html structure

<table>
<tr><th>No.</th><th>Code section</th></tr>
<tr><td>1</td><td>The JavaScript files used are integrated into the HTML document's <head> with the below syntax:

<pre>
<head>
<title data-i18n="title"></title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<!--<script type="text/javascript" src="../file.js"></script>-->
<script type="text/javascript" src="script/jquery-2.1.3.min.js"></script>
<script type="text/javascript" src="script/jquery.cookie.js"></script>
<script type="text/javascript" src="script/i18n_0.0.1.js"></script>
<script type="text/javascript" src="script/Demo10.js"></script>
</head>
</pre>
The "src" property specifies the respective file path.</td></tr>
<tr><td rowspan="2">2</td><td>The web page header is not integrated into the change language option.
It consists of the two structuring classes, "demohead" and "demorow", and the two below text elements with an appropriate formatting instruction.

This page demonstrates: How to realise a language switch.</td></tr>
<tr><td>The texts are directly in the HTML code and are not translated!

<pre>
<body>
    <!-- Head for the Webpage --
    <style>
        .demohead { background-color:#E0E0E0; width:100%}
        .demorow { }
        .democoltitle { width: 189px;font-weight:bold;vertical-align:top;}
        .democoltext   { }
    </style>

    <table class="demohead" >
        <tr class="demorow" >
            <td class="democoltitle" >This page demonstrates:</td>
            <td class="democoltext" >How to realise a language switch.</td>
        </tr>
    </table>
</pre></td></tr>
<tr><td>3</td><td>Changing the language using the flag icons is shown below.
The flag icons are linked to the "language" class.

<pre>

    <img src="Images/flag-deutschland.png"
        class="language" data-lang="de" data-i18n="lang.de"
        width="30px" height="18px" />

</pre>
The "language" class with the (data-lang="..") property is assigned to the change language elements, here, e.g., "German".
The (data-lang) property specifies the language the page is to change to.
Note: For other languages, the code section has the same structure.</td></tr>
</table>

© Siemens AG 2018 All rights reserved

No.	Code section
4	In the same way as the text above the flag icons on the user-defined page, the following HTML text elements are changed between “de” and “en”. **Textfield with HTML - (text.header in en)** Simple text container structured with HTML syntax. - (text.content in de)

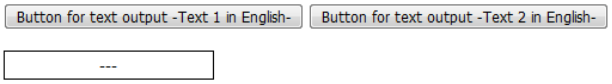
	The “data-i18n” property refers to a uniquely identifiable name, e.g., “text.header” in the “i18n.txt” file.
	`<div class="container" style="clear: left">`
	`  <h1> <span data-i18n="text.header"> </span></h1>`
	`  <span data-i18n="text.content"> </span>`
	`</div>`

	Reference from Demo10.html to “I18n.txt”
	The “i18n.txt” file assigns the texts of the respective language to the identifying names such as (“text.header”).
	Therefore, “text field with HTML...” is displayed on the user-defined page instead of “text.header”.
	`"text.header" : {`
	`  "de": "Textfeld mit HTML - (text.header in de)",`
	`  "en": "Textfield with HTML - (text.header in en)"`
	`},`

© Siemens AG 2018 All rights reserved

10.2.2.2 Demo10.js structure

No.	Code section
1	The script first contains the variable declaration for the functions of the change language elements of the HTML page. `var elements = {};`

2	The following code section implements the assignment of the texts to the buttons and the output field.
	
	`$.init = function(){`
	`  elements.language = $(".language");`
	`  elements.button1 = $("#textByJavaButton1");`
	`  elements.button2 = $("#textByJavaButton2");`
	`  elements.output = $("#textByJava");`
	
	Left: elements.button1
	Right: elements.button2
	Bottom: elements.output
	
	 

3	When the user clicks an element with (class="language"), the (language) property value from "data-lang" is transferred to the "I18n.setLanguage" function in the "i18n_0.0.1.js" script.
	
	Example:
	`elements.language.click(function () {`
	`  I18n.setLanguage( $(this).attr("data-lang") );`
	
	At the same time, the language selection is saved in the "siemens_automation_language" cookie and synchronized with the standard web page.
	
	`$.cookie("siemens_automation_language", $(this).attr("data-lang"),`

	When the user-defined page is opened for the first time, the value of the "I18n.setLanguage" function is applied from the cookie.
	If there is no cookie, the language is set to "en" by default.
	
	`if($.cookie("siemens_automation_language") == null) {`
	`  $.cookie("siemens_automation_language", "en", { path : '/'})`
	`}`
	`I18n.setLanguage($.cookie("siemens_automation_language"));`

4	When clicked, the function outputs the text in the current language based on the identifying name.
	The "data-i18n" property is set to the current ID, "java.output1", so that the text is translated the next time the language is changed.
	
	`elements.button1.click(function () {`
	`  elements.output.html( I18n.translate( "java.output1" ) );`
	`  elements.output.attr("data-i18n","java.output1");`
	`});`

5	Note: For more information about changing texts with JavaScript, please refer to the "Demo10.js" file's code.

© Siemens AG 2018 All rights reserved

10.3 Extending the example

Add a language to the example by duplicating and modifying code sections. Add the identifier for your desired language to the “de”/“en” identifiers – for example, “fr” for French or “xx” as a placeholder for any language.

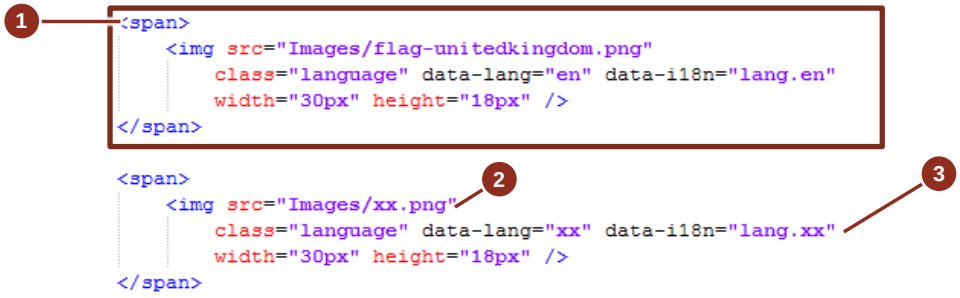
Adding new flag icons

For the new language, you need an appropriate flag icon according to the example, in this case in PNG format. Copy the “xx.png” flag icon to the “…/Images/” folder.

Extending Demo10.html

1. Extend the code section as shown in the figure.
2. In the “Images” path, provide a reference to the new flag icon (xx.png).
3. Change the text assignment for “data-i18n”.

Figure 10-5: Extension in the HTML code

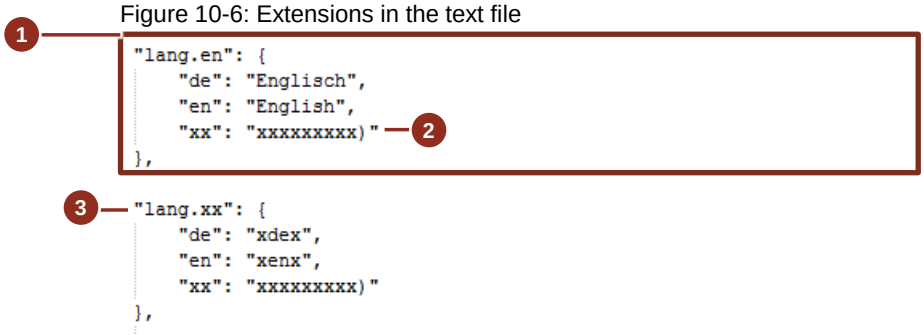


Extending i18n.txt

Add more languages to the “i18n.txt” text file. The text elements for “de” and “en” are assigned to the appropriate names (e.g., “lang.en”). As an example of these names, the following figure shows a code section that must be duplicated if you want to add a new language. Proceed in the same way to extend the text elements.

1. Duplicate the code section.
2. Add the new language item to the code section.
3. Change the name to assign the new code section, in this case “lang.xx”.

Figure 10-6: Extensions in the text file



© Siemens AG 2018 All rights reserved

11 Transferring Data without Reloading Pages Using AJAX

When data is exchanged between a web server and a web browser, web pages are fully reloaded.

This limits the performance of updates that are executed quickly.

Below you will find a definition of the two methods for data transfer that are used in the example.

AJAX

AJAX (“Asynchronous JavaScript and XML”) provides a method for exchanging data asynchronously between a web browser and the web server. Data is exchanged without reloading the web page itself.

Inline frame

Inline frames allow web pages to be split into segments to separate web contents from the surrounding web page.

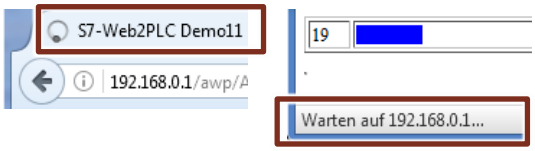
The separated contents are available as independent HTML documents and displayed by the main web page in a defined browser area with a defined size.

Therefore, the contents of the inline frame refresh simultaneously with the surrounding page.

The difference to AJAX becomes clear in Firefox as the tab displays an icon for loading the web page and the bottom status pane displays the “Waiting for192.168.0.1” message.

When using AJAX, these messages are not displayed as the web page itself is not reloaded.

Figure 11-1: Web page refresh in Firefox (inline frame)



© Siemens AG 2018 All rights reserved

11.1 Automation task

The task is to read values using two different methods: “AJAX” and “inline frame”. To this end, two user-defined pages with the same structure have to be programmed that are linked to each other.

Basic function:

- Read the input value for the flow velocity “Velocity”
- Display the level as a bar, depending on the flow velocity.

Requirements for the automation task

- Cyclically load a value with the aid of an inline frame.
- Write a value with the aid of a form.
- Cyclically load a value with the aid of AJAX.
- Write a value with the aid of AJAX.

Requirement for communication between STEP 7 and the user-defined page using AJAX

For the refresh using AJAX to work, integrate the file format with dynamic content in the CPU settings.

Figure 11-2: Requirement – refresh using AJAX with “dat” in STEP 7

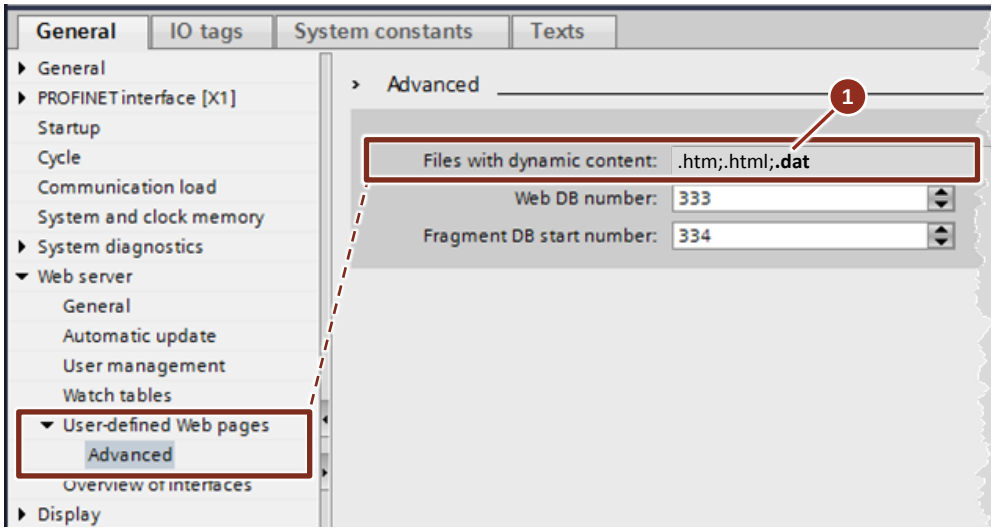


Table 11-1: Instructions for adding a file format

Step	Instruction
1	In the input field, add the “dat” file extension to “.htm;.html;”.

© Siemens AG 2018 All rights reserved

11.2 Functional mechanisms and use

User-defined page structure

On the user-defined page, enter a value for the flow velocity “Velocity”. The blue bar in the bottom display corresponds to the current level. This level increases depending on the flow velocity.

Figure 11-3: User-defined page for reloading pages using AJAX/inline frame

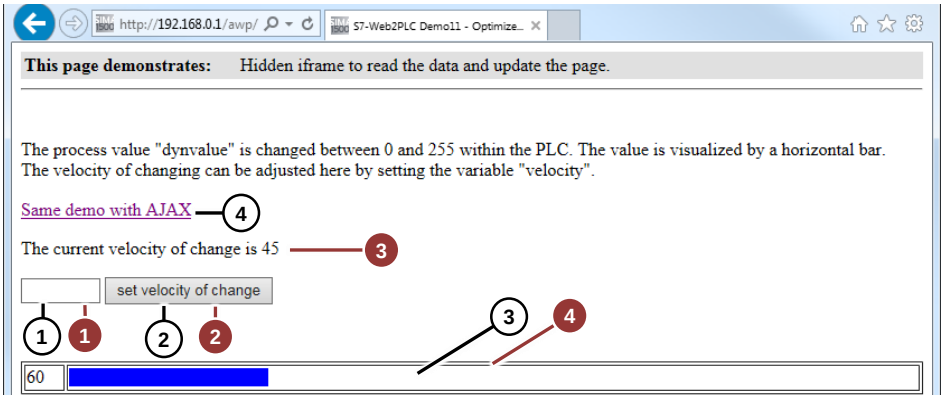

http://192.168.0.1/awp/
S7-Web2PLC Demo11 - Optimize...
This page demonstrates: Hidden iframe to read the data and update the page.
The process value "dynvalue" is changed between 0 and 255 within the PLC. The value is visualized by a horizontal bar.
The velocity of changing can be adjusted here by setting the variable "velocity".
Same demo with AJAX
The current velocity of change is 45
set velocity of change
60
1
2
3
4

Table 11-2: User-defined page for reloading pages using AJAX/inline frame

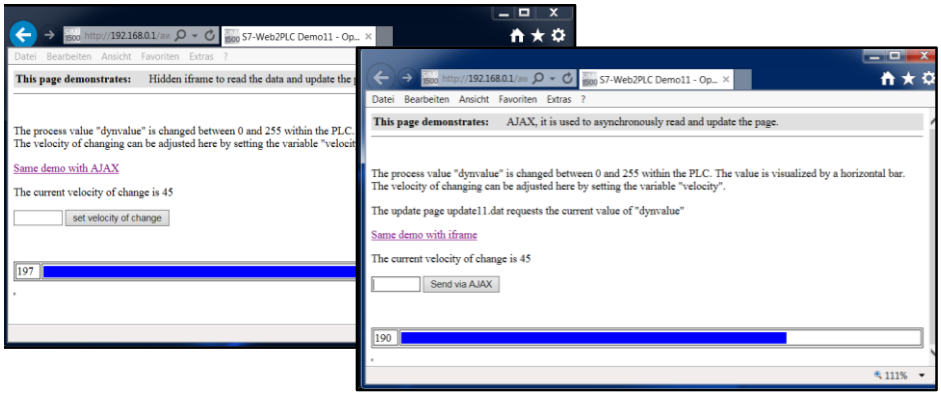
Item	Description
1	This is where you enter new values for the flow velocity “Velocity”.
2	This button transfers the entered value to the controller.
3	Depending on the input value, the bar is filled in blue.
4	This link toggles the user-defined page between inline frame and AJAX.

Table 11-3: Instructions for using the user-defined page

Step	Instruction
1	In the input field, enter a new value.
2	Select the button and the value will be transferred to the CPU.
3	Displays the new value.
4	The bar fills with the velocity depending on the value.

The system displays both methods for transferring the values. It accesses the same variables in the S7 program; as a result, the level display is synchronized on both user-defined pages.

Figure 11-4: Inline frame (left), AJAX (right) user-defined pages

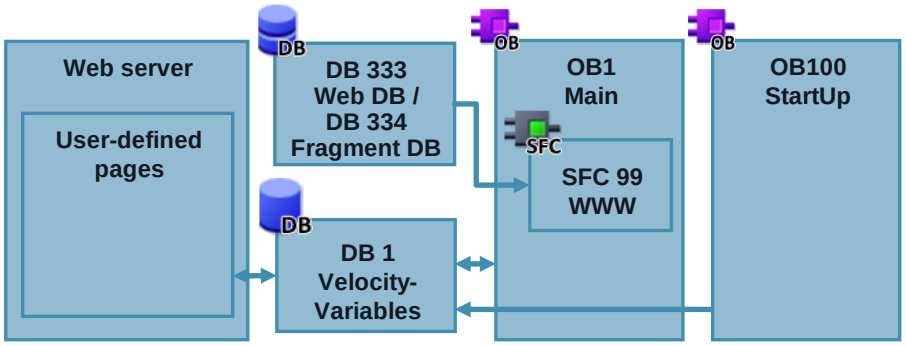

This page demonstrates: Hidden iframe to read the data and update the page.
Same demo with AJAX
The current velocity of change is 45
set velocity of change
197
This page demonstrates: AJAX, it is used to asynchronously read and update the page.
The process value "dynvalue" is changed between 0 and 255 within the PLC. The value is visualized by a horizontal bar.
The velocity of changing can be adjusted here by setting the variable "velocity".
The update page update11.dat requests the current value of "dynvalue"
Same demo with iframe
The current velocity of change is 45
Send via AJAX
190

© Siemens AG 2018 All rights reserved

11.2.1 S7 program structure

A restart resets the variables in DB 1 to their start values. When running OB 1, the user-defined page is refreshed by SFC 99.

Figure 11-5: S7 program for reloading pages using AJAX/inline frame



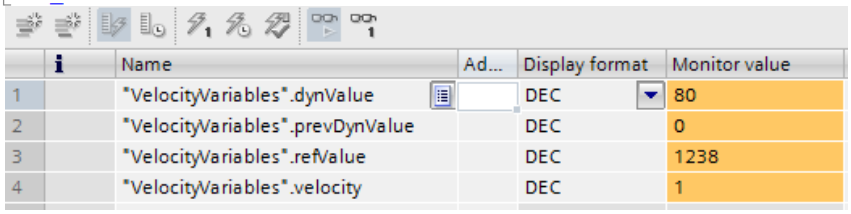
How OB 100 works

No.	Function
1	The “DynValue” and “Velocity” variables are reset in OB100.

	`"VelocityVariables".dynValue := 0;`
	`"VelocityVariables".velocity := 50;`
	• “dynValue” defines the bar level.
	• “Velocity” defines the bar filling speed.

How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

2	Each OB 1 cycle increments the “refValue“” variable by “1”. When the value “2000” is reached, the “dynValue” variable is increased by “1”. This variable of the [Byte] type scales the level display to 255 steps. If the value is increased by 255+1, the “dynValue” variable is reset to “0”.
	`"VelocityVariables".refValue :=`
	`"VelocityVariables".refValue + "VelocityVariables".velocity;`
	`IF "VelocityVariables".refValue >= 2000 THEN`
	`"VelocityVariables".refValue := 0;`
	`"VelocityVariables".dynValue := "VelocityVariables".dynValue + 1;`
	`END_IF;`
	 

Contents of the VelocityVariables DB (DB 1)

DB1 contains the variables for this example.

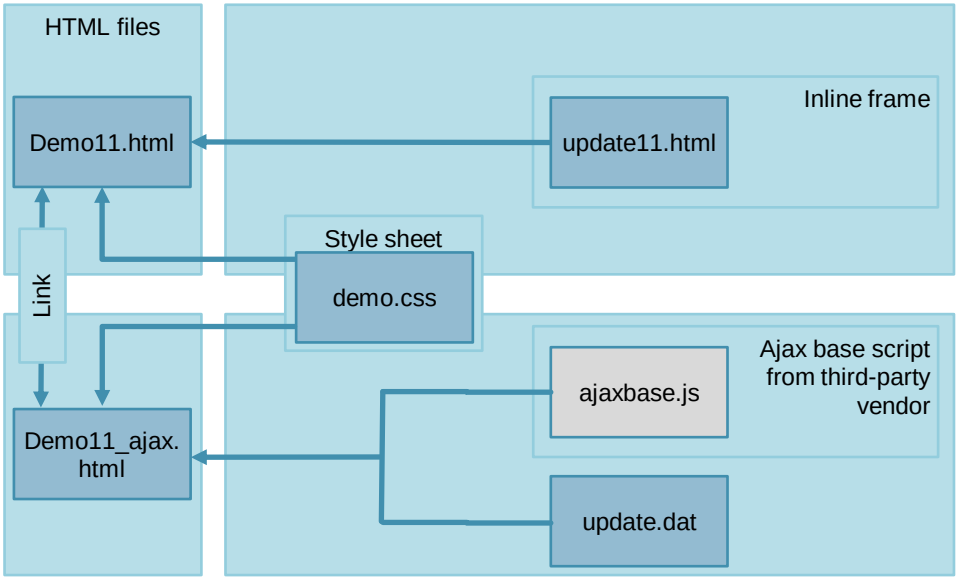
© Siemens AG 2018 All rights reserved

11.2.2 User-defined web page structure

The diagrammatic overview below shows the basic file structure of the user-defined pages.

This description focuses on the JavaScript file's code elements, which makes basic knowledge of HTML necessary for this application example.

Figure 11-6: Diagrammatic overview of the user-defined page



Demo11.html

Defines the structure of the user-defined page and contains the JavaScript functions for updating the S7 variables using the inline frame. The <Head> of the user-defined page contains the JavaScript functions.

update11.html (inline frame for Demo11.html)

The inline frame calls the "ForceUpdate" function from the "demo11.html" file to continuously update the "dynValue" (level) PLC tag.

Demo11_ajax.html

Defines the structure of the user-defined page and contains the JavaScript functions for AJAX in the user-defined page <Head>.

The PLC tags are updated using the JavaScript functions.

update11.dat

Contains the ""dynValue" and "velocity" reference variables that are called from the JavaScript file in Demo11_ajax.html. ("DoHttpRequest" function).

demo.css

Defines the style sheet for the user-defined pages for AJAX and inline frame.

ajaxbase.js

The JavaScript file contains the functions for communication and data exchange between different web browsers and web servers.

© Siemens AG 2018 All rights reserved

11.2.2.1 Demo11.html structure (inline frame)

No.	Demo.html
1	The “Start” function is executed after each page refresh. `<body onload="Start()">`

2	Two functions are called in the “Start” function.
	• “ForceUpdate”
	• “OnTimer” call is delayed by 1000 milliseconds.
	`function Start()`
	`{`
	`  ForceUpdate(:="VelocityVariables".dynValue:);`
	`  setTimeout("OnTimer()",1000);`
	`}`

	“ForceUpdate” determines the bar length from the HTML document and sends it. Formatting for Firefox and IE is done for the bar display.
	`function ForceUpdate(val)`
	`{`
	`  var width, barval;`
	`  var tabelem;`
	`  tabelem = parent.document.getElementById("bar");`
	`  width = tabelem.parentNode.clientWidth;`
	`  barval = ((val*width)/256);`
	`  if (barval == 0) barval = 1;`
	`  tabelem.style.width = Math.floor(barval)+"px";`
	`  var td = parent.document.getElementById("td1");`
	`  if (td.textContent)`
	`  {`
	`  td.textContent = val+"";`
	`  }`
	`  else`
	`  {`
	`  td.innerHTML  = val+"";`
	`  }`
	`  g_bPageRequested = false;`
	`}`

	“OnTimer” updates the value of “DynValue” every 200ms.
	To do this, an inline frame calls the “ForceUpdate” function.
	`function OnTimer()`
	`{`
	`  if (! g_bPageRequested)`
	`  {`
	`  g_bPageRequested = true;`
	`  window.frames["hiddenFrame"].document.location.replace('update11.html');`
	`  }`
	`  setTimeout("OnTimer()", 200);`
	`}`

3	A form that sends values using a “submit” button is used for writing the “Velocity” value.

© Siemens AG 2018 All rights reserved

11.2.2.2 Demo11_ajax.html structure (AJAX)

<table>
<tr><th>No.</th><th>Demo11_ajax.html</th></tr>
<tr><td>1</td><td>The “Start” function is executed after each page refresh.
<code><body onload="Start()"></code></td></tr>
<tr><td>2</td><td>Three functions are called in the “Start” function.
• “DetermineBrowser” determines the browser type (Mozilla, IE11, Chrome, etc.). The function is included in the “ajaxbase.js” file.
“ForceUpdate”; same function as when using the inline frame to determine the bar length.
(See Chapter 11.2.2.1)
• “OnTimer” call is delayed by 1000 milliseconds.

<pre>function Start()
{
    DetermineBrowser();
    ForceUpdate(:="VelocityVariables".dynValue:);
    setTimeout("OnTimer()",1000);
}</pre></td></tr>
<tr><td></td><td>“OnTimer” updates the value of DynValue every 200ms using the “DoHttpRequest” function call from this file: “ajaxbase.js”.
Four values are transferred to the function.
1. The object that has called the current function.
2. The URL “update11.dat” with the variables to be updated.
3. The “UpdateCallback” function that further processes the value and the status code.
4. The “true” value indicates that the data transfer is asynchronous.

When all values have been updated, the “UpdateCallback” function is called.
<pre>function OnTimer()
{
    if (! g_bPageRequested)
    {
        g_bPageRequested = true;

        DoHttpRequest(this, "update11.dat",   UpdateCallback, true);
    }
    setTimeout("OnTimer()", 200);
}</pre></td></tr>
</table>

© Siemens AG 2018 All rights reserved

No.	Demo11_ajax.html
4	The “UpdateCallback” function processes the read values and the status code. The “response” variable contains the updated values of the PLC tags. For processing in a JavaScript file, the values must be split and reassigned. The “status” variable contains the current HTTP status code. If the HTTP status code is less than 300, the “ForceUpdate” function is called and the length of the updated bar is calculated.

```
function UpdateCallback(obj, response, status) {
    var ok;
    var results = response.split(" ");
    var signs = results[0].split("");
    var i;
    var count = 0;
    for (i = 0; i < signs.length; i++) {
        if (true == isNaN(signs[i])) {
            count = count + 1;
        }
        else {break;}
    }
    dynValue = results[0].substr(count, signs.length);

    var dynValueInt = parseInt(dynValue);

    if (status < 300) {
        document.getElementById('veloDiv').innerHTML = results[1],
        ForceUpdate(dynValueInt);
        g_bPageRequested = false;
        setTimeout("OnTimer()", 200);
        return;
    }
    if (status == 503) {
        ok = confirm(dynValueInt);
    } else {
        ok = confirm("FAILED: HTTP error " + status);
    }
    g_bPageRequested = false;
    if (ok) {
        setTimeout("OnTimer()", 1000);
    }
}
```

© Siemens AG 2018 All rights reserved

No.	Demo11_ajax.html
5	Selecting the “Send via AJAX” button calls the “send_ajax_request” function. The current velocity of change is Send via AJAX `<input type="button" onclick="send_ajax_request('%22Velocity%22', 'velocityField')" value="Send via AJAX">`
	The “send_ajax_request” function transfers the entered value to the CPU. `function send_ajax_request(variable, fieldId) { if (window.XMLHttpRequest) { req = new XMLHttpRequest(); } else if (window.ActiveXObject) { req = new ActiveXObject("Microsoft.XMLHTTP"); } else { alert("Der Browser unterstuezt kein Ajax"); } var value = document.getElementById(fieldId).value; var req_url = "?"+variable+"="+value+"&"+Math.random(); //debug alert(req_url); req.open("GET", req_url, false); req.onreadystatechange = ajax_callback; req.send(null); }`

© Siemens AG 2018 All rights reserved

12 Displaying Data of a Datalog as a Graph

12.1 Automation task

The task is to save values of variables to a datalog using the S7 controller.

Datalogs are CSV files that are written to the memory card or the CPU's internal load memory.

The data volume depends on the available memory.

The data from the datalog is to be displayed on a user-defined page as a graph. The data is not updated before the user clicks a button [Load new data].

Note

In this example, sine and cosine values are cyclically written to the SMC or the CPU's memory.

Regarding the read/write cycles, please refer to this entry:

https://support.industry.siemens.com/cs/ww/en/view/109482591

Alternatively, you can program a ring buffer for the values that is used to save a limited number of values and display them in the graph.

© Siemens AG 2018 All rights reserved

Requirements for the automation task

- Cyclically read a datalog.
- Chronologically sort the read data.
- Display the data as a graph.

12.2 Functional mechanisms and use

User-defined page structure

The user-defined page displays the data from the datalog in a line graph. The name of the datalog's CSV file is "SinusUndCosinus.csv".

Figure 12-1: User-defined page for displaying data from a datalog in a line graph

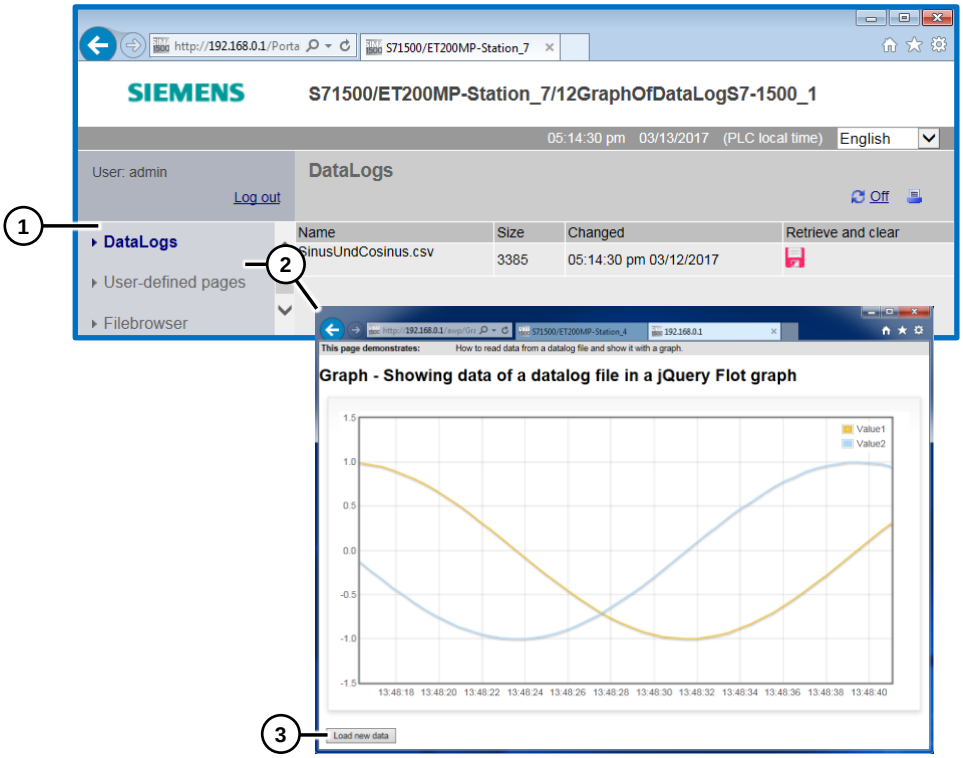


Table 12-1: User-defined page for displaying data from a datalog in a line graph

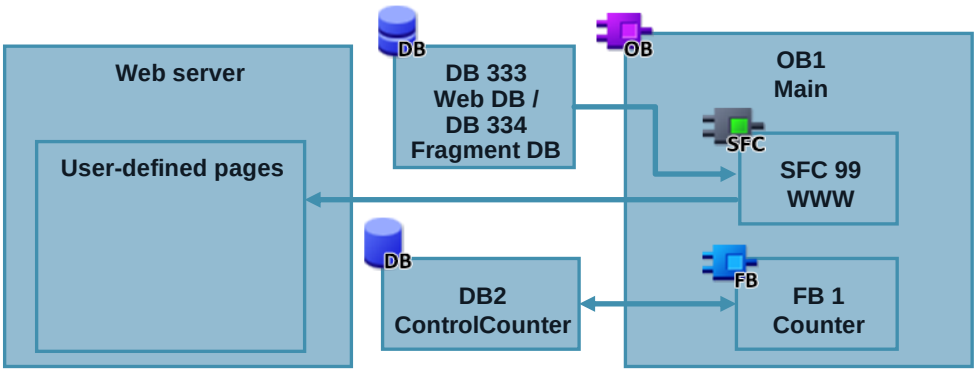
Item	Description
(1)	Datalog stored on memory card as "SinusUndCosinus.csv" file
(2)	Graph displayed on user-defined page (source file: "SinusUndCosinus.csv")
(3)	Button for updating data [Load new data]

© Siemens AG 2018 All rights reserved

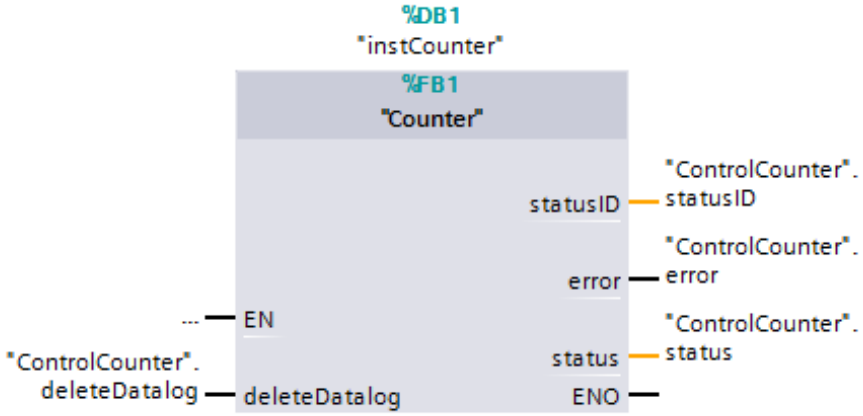
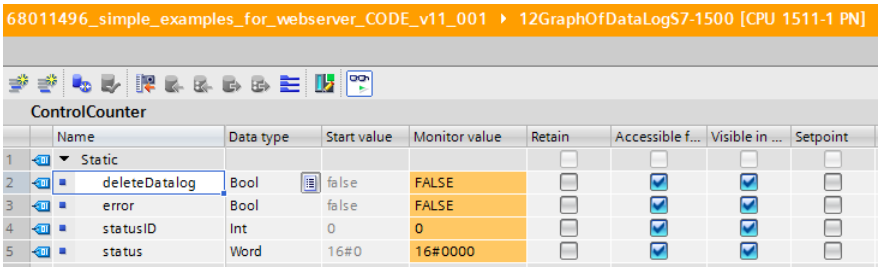
12.2.1 S7 program structure

When running OB 1, the user-defined page is refreshed by SFC 99.

Figure 12-2: S7 program for displaying data from a datalog in a line graph



How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.
2	FB 1 calculates sine and cosine values and creates a datalog. The parameters and name of the datalog CSV file are written to static variables. The sine and cosine values are cyclically saved to this datalog.  The status and error code and the input parameter for deleting the datalog content is stored in DB 2. 

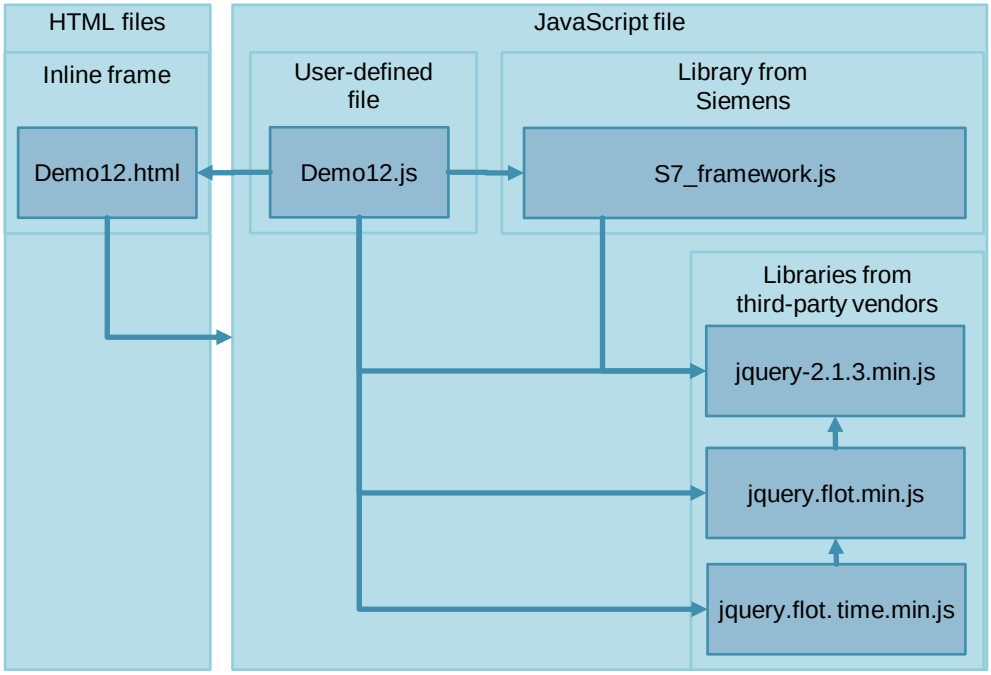
68011496_simple_examples_for_webserver_CODE_v11_001 ▸ 12GraphOfDataLogS7-1500 [CPU 1511-1 PN]
ControlCounter
Name | Data type | Start value | Monitor value | Retain | Accessible f... | Visible in ... | Setpoint
1 Static
2 deleteDatalog Bool false FALSE
3 error Bool false FALSE
4 statusID Int 0 0
5 status Word 16#0 16#0000

© Siemens AG 2018 All rights reserved

12.2.2 User-defined web page structure

The diagrammatic overview below shows the basic file structure of the example.

Figure 12-3: Diagrammatic overview of the user-defined page



© Siemens AG 2018 All rights reserved

Demo12.html with (graph container)

Defines the structure of the user-defined page and includes a container for displaying the graph. The JavaScript files are integrated into the HTML document to display the graph.

Figure 12-4: Demo12.html with graph

Table 12-2: Demo12.html with graph

Item	Description
1	Demo12.html
2	Graph container with access to the datalog data

Demo.js

The “Demo12.js” file calls functions from the different libraries to create a graph from the data of a datalog (CSV file). For information about processing the data, please refer to the source code.

jquery.flot.min.js and jquery.flot.time.min.js (graph template)

These JavaScript files calculate the graph in the inline frame display.
The programming used elements from the jQuery library.

For more information, visit: http://www.flotcharts.org/

S7_framework.js

The JavaScript file contains a number of functions for converting and processing data from a SIMATIC S7.
For more information about the S7 Framework, please refer to Chapter 4.3.

S7_framework.css

Defines the style sheet for the HTML user-defined pages

jquery-2.1.3.min.js (jQuery library)

The jQuery library file allows more efficient programming with JavaScript. For more information about the jQuery library, please refer to Chapter 4.3.

© Siemens AG 2018 All rights reserved

12.2.2.1 Demo12.html structure

No.	Demo12.html code section

1	The JavaScript files used are integrated into the HTML document's <head> with the below syntax:
	
	`<head>`
	`<meta http-equiv="content-type" content="text/html; charset=UTF-8">`
	`<link rel="stylesheet" type="text/css" href="CSS/S7_framework.css">`
	`<script type="text/javascript" src="script/jquery-2.1.3.min.js"></scri`
	`<script type="text/javascript" src="script/jquery.flot.min.js"></scrip`
	`<script type="text/javascript" src="script/jquery.flot.time.min.js"></`
	`<script type="text/javascript" src="script/S7_framework_0.1.7.js"></sc`
	`<script type="text/javascript" src="script/Demo12.js"></script>`
	`<script type="text/javascript"></script>`
	`</head>`
	
	The "src" property specifies the respective file path. User-defined pages cannot directly access a datalog.

2	The datalog CSV files are stored on the standard web page that must be opened for access.
	
	The graph is called in a container element.
	
	`<h1><span>Graph - Showing data of a datalog file in a jQuery Flot graph</span></h1>`
	`<div class="graph-container">`
	`<div id="graph-placeholder"></div>`
	`</div>`
	
	The "graph-container" "div" tag is a placeholder for the graph.
	The "Demo12.js" JavaScript file composes the graph in the placeholder.

© Siemens AG 2018 All rights reserved

12.2.2.2 Demo12.js structure

<table>
<tr><th>No.</th><th>Code section</th></tr>
<tr><td>1</td><td>Determining the CPU type (S7-1200 or S7-1500)
In the “Demo12.js” JavaScript file, a function to determine whether an S7-1200 or S7-1500 is used is executed in “$(document).ready(function(){}”.
<pre>
$(document).ready(function(){
    if (plcType != "1200" && plcType != "1500")
    {
        $.ajax({type:"GET",url:"../../../Portal/Intro.mwsl",data:"",dataType:"text"})
                .done(function(webpageData){
                    var search12 = webpageData.search("CPU12");
                    var search15 = webpageData.search("CPU15");
                    if (search12 >= 0)
                    {
                        plcType = "1200";
                        $.init();
                    }
                    else if (search15 >= 0)
                    {
                        plcType = "1500";
                        $.init();
</pre>
Alternatively, you can directly call the functions of “$.init” in “$(document).ready(function[]{}”.
Replace “plcType” with “1200” or “1500”, including quotation marks.</td></tr>
<tr><td></td><td>The PLC type is transferred to the S7 Framework with the following function.
<pre>
$.init = function(){
    S7Framework.initialize(plcType, "");
    S7Framework.readDataLog("SinusUndCosinus","Read Datalog failed", decodeCSV);
}
</pre>
Alternatively, write “1200” or “1500” for plcType.
Example: S7Framework.initalize(“1200”, “”);</td></tr>
<tr><td>2</td><td>Datalog (read values)

The “readDataLog” function reads datalogs.
<pre>S7Framework.readDataLog("SinusUndCosinus","Read Datalog Failed",decodeCSV);</pre>
The name of the “SinusUndCosinus” datalog has to be transferred to the function.
Text as an error message is optional.

The function name of “decodeCSV” is the destination for the read values.</td></tr>
<tr><td>3</td><td>Datalog (process values)

The “decodeCSV” function processes the values from the datalog.
<pre>function decodeCSV(CSVdata){</pre>
All time values are converted to [Date.UTC] format and sorted chronologically.
The data is prepared for the graph template and evaluated in the JavaScript file (jquery.flot.time.min.js).
This function also defines the graph properties.
(For example, connect points, no steps in graph.)</td></tr>
</table>

© Siemens AG 2018 All rights reserved

13 Display Elements Using Scalable Vector Graphics

13.1 Programming task

The task is to create a selection of display elements using Scalable Vector Graphics (SVG). HTML5-capable browsers support this technique.

Requirements for the automation task

- Create SVG elements
- Dynamize SVG elements using JavaScript

13.2 Functional mechanisms and use

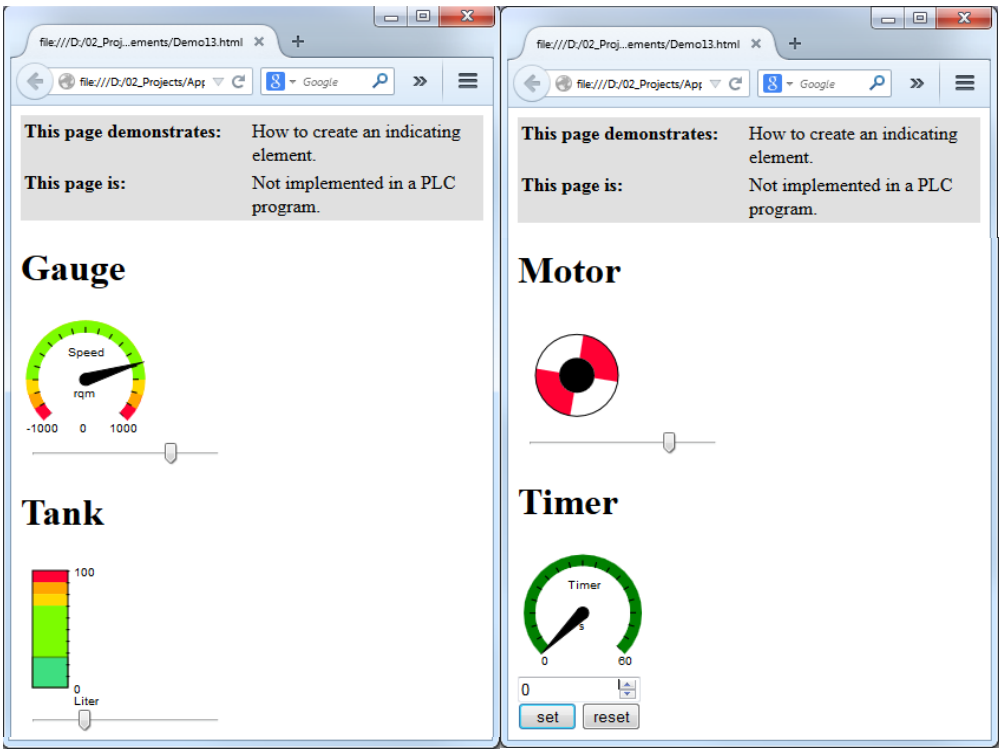
Note

This example does not contain a specific S7 program. The display elements do not explicitly access data of a SIMATIC S7.
You will find the HTML file in the project folder in “html_indicatingElements”.

User-defined page structure

The user-defined page shows different display elements that can be modified and controlled using a slider below the display elements. For connection to an S7, replace the slider with the appropriate UDP variables.

Figure 13-1: User-defined page with SVG (Scalable Vector Graphics) display elements



© Siemens AG 2018 All rights reserved

13.2.1 Sample SVG element – rotary motion of a motor

Programming SVG elements

Define the shape of SVG display elements by the HTML code. Program a JavaScript file for the motion functions. In general, use the input variables to change the position/rotation of circular/rectangle elements.

The input variables are implemented by sliders below the SVG elements.

The following figure shows an example of a motor's rotary motion and the associated code section in the HTML document. The motor is composed of a total of three circular elements.
The rotary motion is implemented by a dash line (white, red, white, red) that rotates inside a second circle for the outer boundary. The inner circle represents the rotor's cam.

You can scale the motor by changing the values in quotation marks. Always change the value for the dash line such that the circle's circumference exactly matches 4 dashes. The specific parameter for this is called "stroke-dasharray".

Figure 13-2: Sample SVG display element – motor design

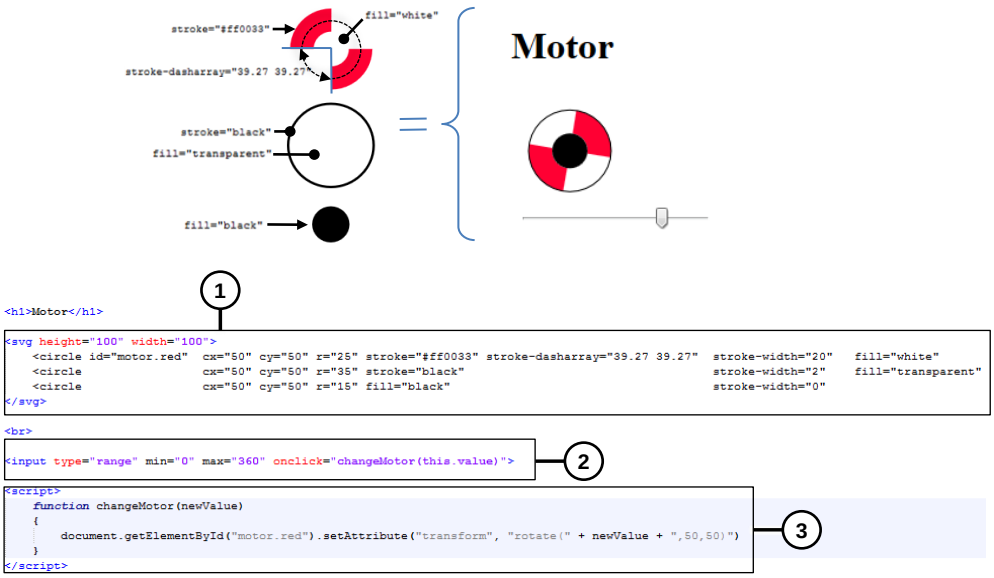


Table 13-1: Motor design display elements

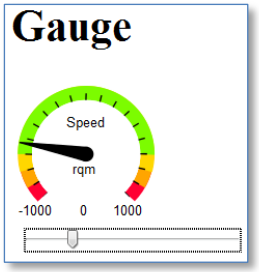
Item	Description
①	Circles: Define position, size and colors for three circular elements.
②	Slider: Graphic element for entering values.
③	Function: rotate motor

Note

To extend your application, copy the SVG element's code section directly to your HTML document. If necessary, replace the input variable of the slider element as required for your application.

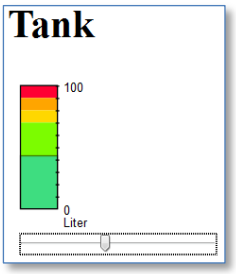
© Siemens AG 2018 All rights reserved

13.2.1.1 Demo13.html structure



No.	Code sections from (Gauge)
1	**<svg width="110" height="110">** **…</svg>** Der "svg" tag defines the graphic area. The graphic area starts at the top left and the size of the coordinate axes corresponds to the graphics size in pixels. In the "svg" tag, you can create the following elements:
2	**<circle cx="55" cy="55" r="40" />** The "circle" tag creates a circle. The "cx" property specifies the position of the center point on the X axis, "cy" defines the corresponding position on the Y axis. The "r" property specifies the radius.
3	**stroke="#ff0033"** The "stroke" property specifies the border color/line color of SVG elements.
4	**stroke-width="20"** The "stroke-width" property specifies the border thickness/line thickness of SVG elements.
5	**stroke-dasharray="188.25 63.08"** The "stroke-dasharray" property deletes parts of the border or a line of SVG elements. The first value specifies the length to be shown. The second value specifies the length to be deleted. The two lengths alternate on the border/line until they have circled around the border/reached the end of the line. As a result, you get a dash border/line. Note: For circles, the border is the circumference. It is calculated as follows: 2 * π * r.
6	**fill="white"** The "fill" property specifies the fill color or font color of the SVG element.
7	**transform="rotate(135, 55, 55)"** The "transform rotate" property rotates SVG elements. The first number is the number of degrees and specifies how far the element will be rotated. The second and third number specify the point around which the element will be rotated.
8	**id="gauge.pointer"** The "id" property assigns an SVG element an ID to be accessed using JavaScript.

© Siemens AG 2018 All rights reserved



No.	Code section from (Tank)
1	**<rect x="10" y="10" width="30" height="10" />** The "rect" tag creates a rectangle. The "x" property specifies the top left point of the rectangle on the X axis, "y" defines the corresponding point on the Y axis. The "width" property specifies the width and "height" defines the height of the rectangle.
2	**<line x1="40" y1="10" x2="40" y2="110" />** The "line" tag creates a line. The "x1" and "y1" properties specify the line's start point. The "x2" and "y2" properties specify the line's end point.
3	**<polygon points="50,55 60,55 55,5" />** The "polygon" tag creates a polygon. The "points" property specifies the polygon's corners.
4	**<text x="40" y="35" font-size="10" font-family="Helvetica, Arial, sans-serif">** **Speed** **</text>** The "text" tag creates text. The "x" property specifies the position of the bottom right point of the text field on the X axis, "y" defines the corresponding point on the Y axis. The "font-size" property specifies the font size. The "font-family" property specifies the font. The desired text is written to this tag.
No.	**Code section (structure of a JavaScript script)**
1	**<script>** **var newValue** **…** **document.getElementById("gauge.pointer").** **setAttribute("transform","rotate("+newValue+",55,55)")** **</script>** **"document.getElementById()"** Specifies the element to be accessed or modified. **"setAttribute"** The function changes the property of elements. Elements that do not represent JavaScript variables are put in quotation marks. Prior to executing the function, JavaScript variables are replaced with the variable value.

Note

For more information about creating SVG elements, visit:

http://www.w3schools.com/html/html5_svg.asp
http://www.w3schools.com/graphics/svg_intro.asp
http://wiki.selfhtml.org/wiki/SVG

© Siemens AG 2018 All rights reserved

14 S7 Diagnostics and Loading Indicator

14.1 Automation task

On the standard web page, the S7 web server provides the option to read the CPU's diagnostic buffer.

In practice, one possible requirement could be to display the diagnostic buffer in conjunction with other information on a user-defined page.

The task is to program a user-defined page that displays the diagnostic buffer of the S7 controller. Loading is to be indicated by an animated icon as a loading indicator.

Requirements for the automation task

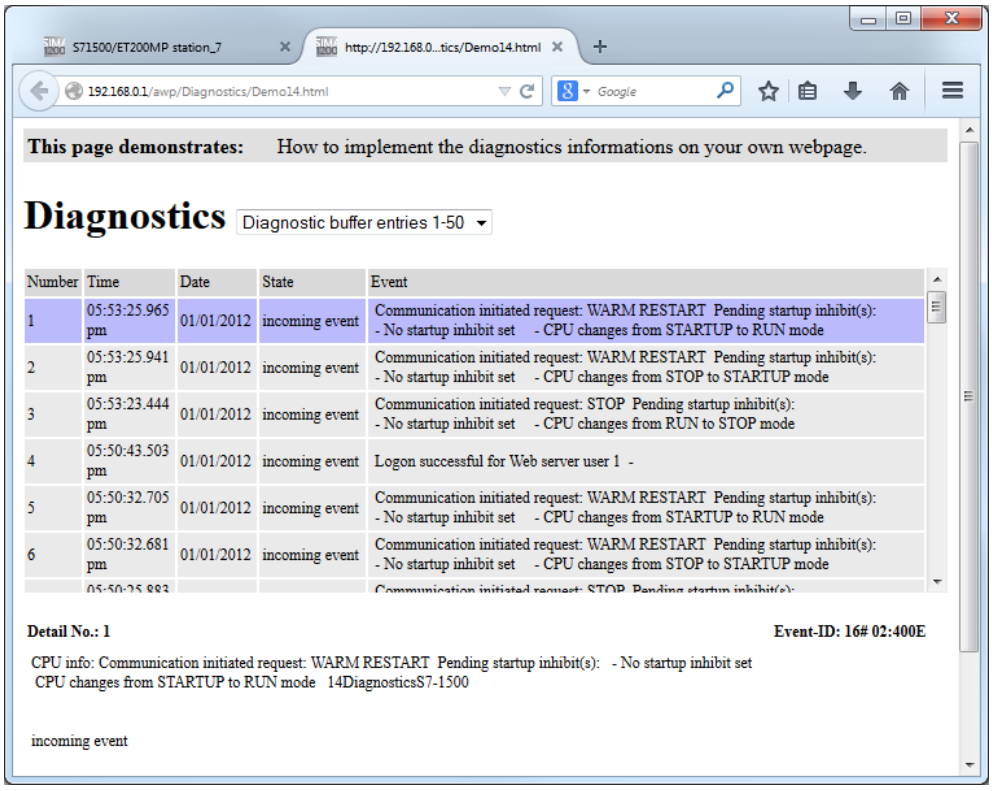
- Display diagnostic information of the S7 controller on a user-defined page
- Integrate a loading indicator with the S7 Framework

14.2 Functional mechanisms and use

User-defined page structure

The user-defined page displays the diagnostic information of the CPU. Clicking a diagnostic message displays detailed information about this message below the table.

Figure 14-1: User-defined page with diagnostic data

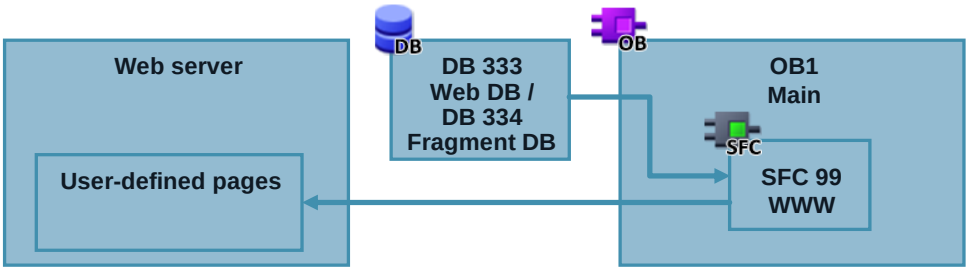

S71500/ET200MP station_7
http://192.168.0...tics/Demo14.html
192.168.0.1/awp/Diagnostics/Demo14.html
Google
This page demonstrates: How to implement the diagnostics informations on your own webpage.
Diagnostics
Diagnostic buffer entries 1-50
Number Time Date State Event
1 05:53:25.965 pm 01/01/2012 incoming event Communication initiated request: WARM RESTART Pending startup inhibit(s): - No startup inhibit set - CPU changes from STARTUP to RUN mode
2 05:53:25.941 pm 01/01/2012 incoming event Communication initiated request: WARM RESTART Pending startup inhibit(s): - No startup inhibit set - CPU changes from STOP to STARTUP mode
3 05:53:23.444 pm 01/01/2012 incoming event Communication initiated request: STOP Pending startup inhibit(s): - No startup inhibit set - CPU changes from RUN to STOP mode
4 05:50:43.503 pm 01/01/2012 incoming event Logon successful for Web server user 1 -
5 05:50:32.705 pm 01/01/2012 incoming event Communication initiated request: WARM RESTART Pending startup inhibit(s): - No startup inhibit set - CPU changes from STARTUP to RUN mode
6 05:50:32.681 pm 01/01/2012 incoming event Communication initiated request: WARM RESTART Pending startup inhibit(s): - No startup inhibit set - CPU changes from STOP to STARTUP mode
Detail No.: 1
Event-ID: 16# 02:400E
CPU info: Communication initiated request: WARM RESTART Pending startup inhibit(s): - No startup inhibit set
CPU changes from STARTUP to RUN mode 14DiagnosticsS7-1500
incoming event

© Siemens AG 2018 All rights reserved

14.2.1 S7 program structure

In the S7 program, only SFC 99 is called.

Figure 14-2: S7 program for the user-defined page with diagnostic data and loading indicator



How OB 1 works

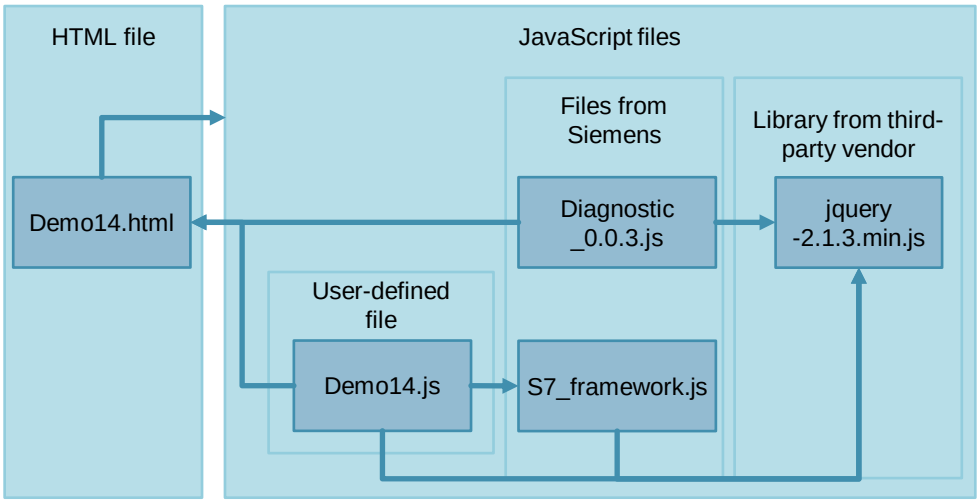
No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

14.2.2 User-defined web page structure

The diagrammatic overview below shows the basic structure of the user-defined page.

User-defined page structure

Figure 14-3: Diagrammatic overview of the user-defined page



© Siemens AG 2018 All rights reserved

Demo14.html

The HTML document defines the structure of the user-defined page and the calls of the JavaScript files used, see Chapter 14.2.2.1. The document defines that clicking the diagnostic messages displays them in a table.

Demo14.js

The JavaScript file integrates the S7 Framework to start the loading indicator.

diagnostic_0.0.1.js

The “diagnostic_0.0.1.js” file accesses tags with a defined ID.
The diagnostic display is inserted into these tags.

S7_framework.js

The JavaScript file contains a number of functions for converting and processing data from a SIMATIC S7.
For more information about the S7 Framework, please refer to Chapter 4.3.

jquery-2.1.3.min.js (jQuery library)

The jQuery library file allows more efficient programming with JavaScript. For more information about the jQuery library, please refer to Chapter 4.3.

diag.css (style sheet)

Style sheet for the loading indicator. This style sheet defines the position and web animation for the “…\img\sprites.png” file.

S7_diagnostic.css (style sheet)

Style sheet for designing the table with the diagnostic information.

Load / Sprites (images)

The loading indicator is an animated file in GIF format that accesses the graphic elements of a PNG file as shown below.

Figure 14-4: Designing image for load indicator

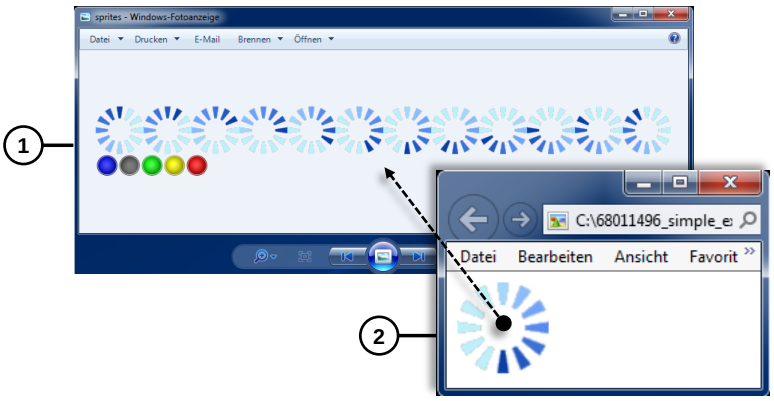


Table 14-1: Diagrammatic overview of the user-defined page

Item	Description
1	PNG source file with 12 single images, “sprites.png”
2	GIF target file with time change of the single images (animation), “load.gif”

© Siemens AG 2018 All rights reserved

14.2.2.1 Demo14.html structure

No.	Code description

1	The style sheets and JavaScript files used are integrated into the HTML document's <head>:
	Style sheets: <link rel=…>
	JavaScript files: <script type=…>
	
	`<head>`
	`  <meta http-equiv="content-type" content="text/html; charset=UTF-8">`
	`  <link rel="stylesheet" type="text/css" href="CSS/S7_diagnostic.css">`
	`  <link rel="stylesheet" type="text/css" href="CSS/diag.css">`
	`  <script type="text/javascript" src="script/jquery-2.1.3.min.js"></script>`
	`  <script type="text/javascript" src="script/diagnostic_0.0.1.js"></script>`
	`  <script type="text/javascript" src="script/S7_framework_0.1.7.js"></script>`
	`  <script type="text/javascript" src="script/Demo14.js"></script>`
	`  <script type="text/javascript"></script>`
	`</head>`
	
	The "src" property specifies the respective file path.

2	The loading indicator is displayed in this "div" tag.
	
	`<body>`
	`  <div id="loading_div">`
	`  <div class="loading_pic"></div>`
	`  </div>`

3	In this "div" tag, the diagnostic information is displayed as a table. `<div id="diagTableDiv" width="1000px" height="250px"></div> `
4	In this "div" tag, the "Event" information of a diagnostic message is displayed in the table when the appropriate message is clicked. `<div id="diagDetailDiv" width="1000px" height="250px">` `  <span id="diagDetailNo"></span>` `  <span id="diagDetailEventID"></span> ` `  <span id="diagDetailText">DiagnosticDetail</span>` `</div>`

© Siemens AG 2018 All rights reserved

14.2.2.2 Demo14.js structure

No.	Code description

1	**Determining the CPU type (S7-1200 or S7-1500)**
	In the “Demo12.js” JavaScript file, a function to determine whether an S7-1200 or S7-1500 is used is executed in “$(document).ready(function(){}”.
	`$(document).ready(function(){`
	`  if (plcType != "1200" && plcType != "1500")`
	`  {`
	`  $.ajax({type:"GET",url:"../../../Portal/Intro.mwsl",data:"",dataType:"text"})`
	`  .done(function(webpageData){`
	`  var search12 = webpageData.search("CPU12");`
	`  var search15 = webpageData.search("CPU15");`
	`  if (search12 >= 0)`
	`  {`
	`  plcType = "1200";`
	`  $.init();`
	`  }`
	`  else if (search15 >= 0)`
	`  {`
	`  plcType = "1500";`
	`  $.init();`
	
	Alternatively, you can directly call the functions of “$.init” in “$(document).ready(function[]{}”.
	Replace “plcType” with “1200” or “1500”, including quotation marks.

2	This function transfers the CPU type and the ID of the “div” tag where the loading indicator is to be displayed to the S7 Framework.
	The S7 Framework detects when an AJAX function is executed and shows/hides the loading indicator accordingly.
	
	`$.init = function(){`
	`  S7Framework.initialize(plcType, "#loading_div");`

	This function runs the script from diagnostic_0.0.3.js. It inserts the diagnostic information into the “div” tags of the HTML file. The “div” tags are identified by the appropriate IDs. `diagnose.initialize(plcType);`

	The loading indicator is shown when the “div” tag with ID “diagTableDiv” is empty.
	The following query is programmed for this.
	
	`if ($("#diagTableDiv") == "[object Object]"){`
	`  $("#loading_div").show();`
	`}else{`
	`  $("#loading_div").hide();`
	`}`

© Siemens AG 2018 All rights reserved

14.2.2.3 diag.css structure

No.	Code description
1	The “sprites.png” image for the loading indicator is in the “img” folder. The “diag.css” file references the sprites.png image as shown below.

```
.loading_pic{
position: absolute;
width: 64px;
height: 64px;
top: 50%;
left: 50%;
margin-top: -32px;
margin-left: -32px;
background-image: url(../img/sprites.png);
-webkit-animation: play .8s steps(12) infinite;
   -moz-animation: play .8s steps(12) infinite;
    -ms-animation: play .8s steps(12) infinite;
     -o-animation: play .8s steps(12) infinite;
        animation: play .8s steps(12) infinite;
        animation-direction:reverse;
}
```

© Siemens AG 2018 All rights reserved

15 Changing Pictures Using JavaScript

15.1 Programming task

The task is to change a picture depending on a JavaScript variable.

Requirements for the automation task

- Display pictures depending on a JavaScript variable

15.2 Functional mechanisms and use

Note This example has no S7 program. You will find the HTML file in the project folder in “html_pictureChange”.

User-defined page structure

The user-defined page displays a picture with two buttons below it.

Figure 15-1: User-defined page for changing pictures using a JavaScript variable

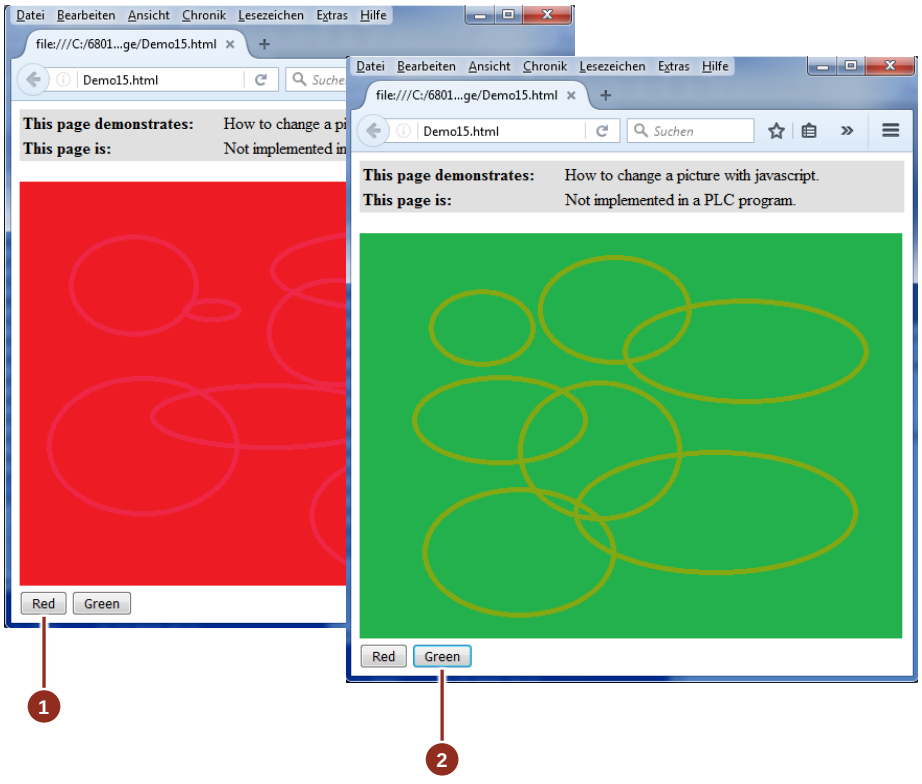


Table 15-1: Instructions for using the user-defined page

Step	Instruction
1	Select the “Red” button and the red picture will be displayed.
2	Select the “Green” button and the green picture will be displayed.

© Siemens AG 2018 All rights reserved

15.2.1.1 Demo15.html structure

Creating SVG elements

No.	Code description
1	With this HTML element, the red picture is displayed when starting the user-defined page. `<img id="pic" src="img\red.png">`

2	With these HTML elements, one button labeled “Red” and one button labeled “Green” are displayed.
	
	Clicking the “Green” button triggers the “changePic” JavaScript function and transfers the “green” value to it. In the JavaScript file, the parameter (color) therefore includes the value (green).
	
	`<button onclick="changePic('red')">Red</button>`
	`<button onclick="changePic('green')">Green</button>`

3	The “changePic” function is defined in a JavaScript part of the HTML file.
	
	To change a picture depending on a PLC tag, the PLC tag must be read cyclically. Cyclic reading is done by the IF statement.
	
	When the “green” value has been transferred to the function, the previous path to the red picture is changed to the path of the green picture.
	
	As a result, the green picture is displayed.
	`<script>`
	`function changePic(color)`
	`{`
	`if (color == "green")`
	`{`
	`document.getElementById("pic").src = "img/green.png";`
	`}`
	`else`
	`{`
	`document.getElementById("pic").src = "img/red.png";`
	`}`
	`}`
	`</script>`

© Siemens AG 2018 All rights reserved

16 Button for CPU Restart

16.1 Automation task

The task is to stop and then automatically restart the CPU using a button.

Requirements for the automation task

- Implement a button for restarting the CPU

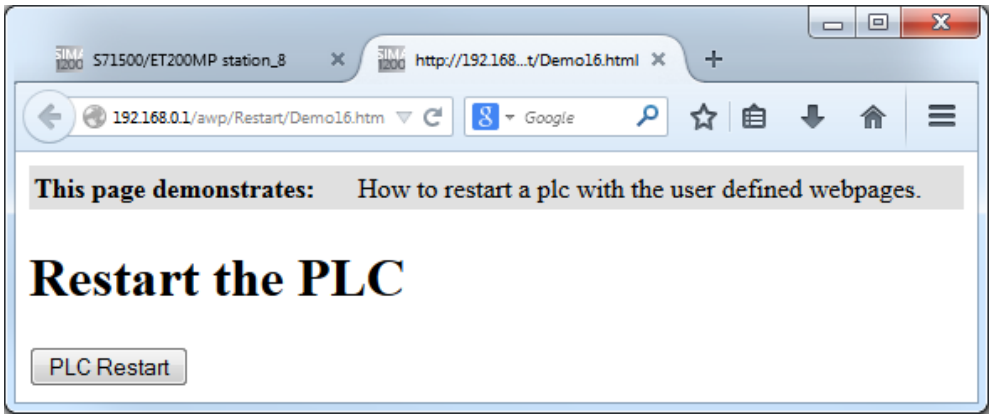
16.2 Functional mechanisms and use

User-defined page structure

Selecting the “PLC Restart” button stops and then automatically restarts the CPU.

Technically, this is a warm restart of the CPU. Program execution restarts and retentive data is retained.

Figure 16-1: User-defined page with button for CPU restart

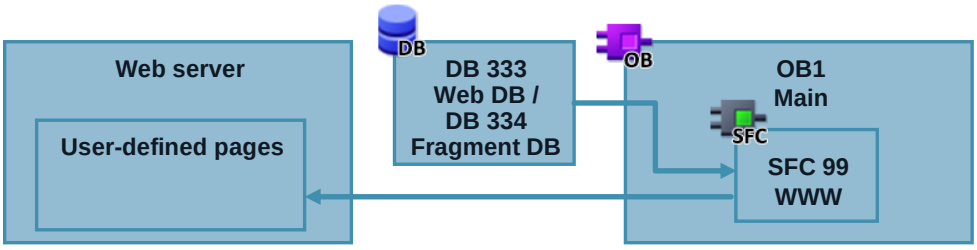


© Siemens AG 2018 All rights reserved

16.2.1 S7 program structure

In the S7 program, only SFC 99 is called.

Figure 16-2: S7 program for user-defined page with button for CPU restart



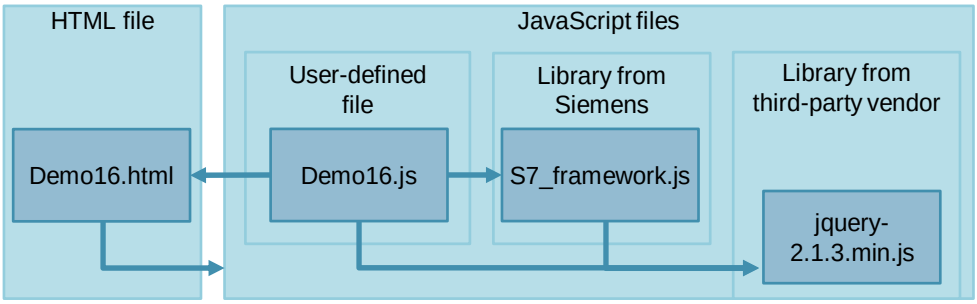
How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

16.2.2 User-defined web page structure

The diagrammatic overview below shows the basic file structure of the user-defined page.

Figure 16-3: Diagrammatic overview of the user-defined page



Demo16.html

In the “Demo16.html” file, all JavaScript files are integrated and the button is defined.

Demo16.js

The JavaScript file performs the restart with the aid of “S7_framework.js”.

S7_framework.js

The JavaScript file contains a number of functions for converting and processing data from a SIMATIC S7.
For more information about the S7 Framework, please refer to Chapter 4.3.

jquery-2.1.3.min.js (jQuery library)

The jQuery library file allows more efficient programming with JavaScript. For more information about the jQuery library, please refer to Chapter 4.3.

© Siemens AG 2018 All rights reserved

16.2.2.1 Demo16.html structure

No.	Code description

1	The JavaScript files used are integrated into the HTML document's <head>:
	
	`<head>`
	`<meta http-equiv="content-type" content="text/html; charset=UTF-8">`
	`<script type="text/javascript" src="script/jquery-2.1.3.min.js"></script>`
	`<script type="text/javascript" src="script/S7_framework_0.1.7.js"></script>`
	`<script type="text/javascript" src="script/Demo16.js"></script>`
	`</head>`
	
	The "src" property specifies the respective file path.

2	Create a button named "PLC Restart".
	
	`<div class="container">`
	`<h1><span>Restart the PLC</span></h1>`
	`<button id="restart">PLC Restart</button>`
	`</div>`

16.2.2.2 Demo16.js structure

No.	Code description

1	**Determining the CPU type (S7-1200 or S7-1500)**
	In the "Demo12.js" JavaScript file, a function to determine whether an S7-1200 or S7-1500 is used is executed in "$(document).ready(function(){}".
	`$(document).ready(function(){`
	`  if (plcType != "1200" && plcType != "1500")`
	`  {`
	`  $.ajax({type:"GET",url:"../../../Portal/Intro.mwsl",data:"",dataType:"text"})`
	`  .done(function(webpageData){`
	`  var search12 = webpageData.search("CPU12");`
	`  var search15 = webpageData.search("CPU15");`
	`  if (search12 >= 0)`
	`  {`
	`  plcType = "1200";`
	`  $.init();`
	`  }`
	`  else if (search15 >= 0)`
	`  {`
	`  plcType = "1500";`
	`  $.init();`
	
	Alternatively, you can directly call the functions of "$.init" in "$(document).ready(function[]{}".
	Replace "plcType" with "1200" or "1500", including quotation marks.

2	The "S7Framework.initialize(plcType,"")" function transfers the CPU type to the S7 Framework.
	Selecting the button with the "restart" ID restarts the CPU using the "S7Framework.restartCPU();" function.
	
	`$.init = function(){`
	`  S7Framework.initialize(plcType, "");`
	`  $("#restart").click(function(){`
	`  S7Framework.restartCPU();`
	`  });`
	`}`

© Siemens AG 2018 All rights reserved

17 Login on User-Defined Pages

17.1 Automation task

The task is to display the user-defined pages via a separate login window.

Requirements for the automation task

- Integrate the login window
- Edit the style properties of the login window

17.2 Functional mechanisms and use

User-defined page structure

On the user-defined page, you log in using the input fields and the “Log in” button. Use the login information listed in Chapter 4.1.

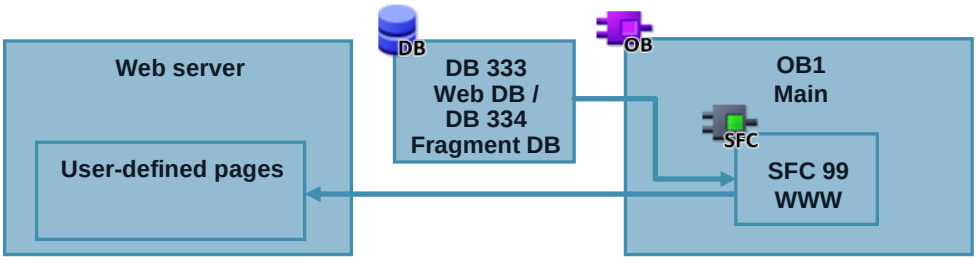
Figure 17-1: Login on user-defined pages

© Siemens AG 2018 All rights reserved

17.2.1 S7 program structure

In the S7 program, only SFC 99 is called.

Figure 17-2: S7 program for login on the user-defined page



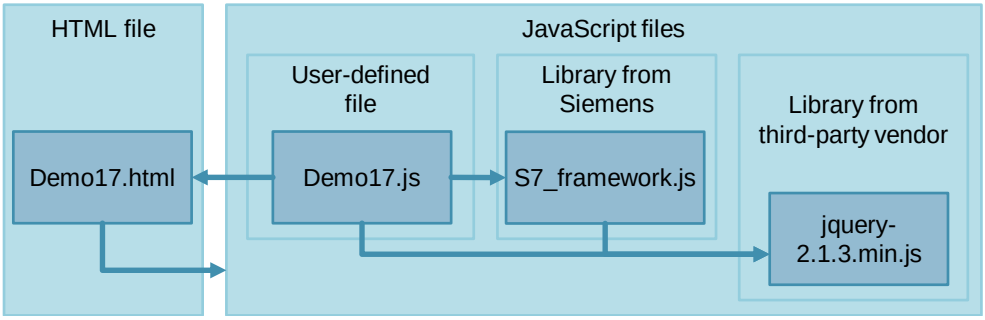
How OB 1 works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.

17.2.2 User-defined page structure

The diagrammatic overview below shows the basic file structure of the user-defined page.

Figure 17-3: Diagrammatic overview of the user-defined page



Demo17.html

All JavaScript files are integrated into the “Demo17.html” file.

Demo17.js

The JavaScript file allows the user to log in and out with the aid of “S7_framework.js”.

S7_framework.js

The JavaScript file contains a number of functions for converting and processing data from a SIMATIC S7.
For more information about the S7 Framework, please refer to Chapter 4.3.

jquery-2.1.3.min.js (jQuery library)

The jQuery library file allows more efficient programming with JavaScript. For more information about the jQuery library, please refer to Chapter 4.3.

© Siemens AG 2018 All rights reserved

17.2.2.1 Demo17.html structure

No.	Code description

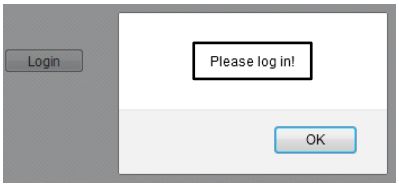
1	The JavaScript files used are integrated into the HTML document's <head> with the below syntax:
	
	`<head>`
	`  <title>Log In</title>`
	`  <meta charset="UTF-8">`
	`  <script type="text/javascript" src="script/jquery-2.1.3.min.js"></script>`
	`  <script type="text/javascript" src="script/S7_framework_0.1.7.js"></script>`
	`  <script type="text/javascript" src="script/Demo17.js"></script>`
	`</head>`
	
	The "src" property specifies the respective file path.

2	The "login" "div" box defines the login area as an inline frame with the "src=…" source from the standard web page.
	Using the JavaScript file, the login window is inserted into the "loginBox" "div" box.
	
	`<h1>Log in at the user defined webpages</h1>`
	`<div id="login">`
	`  <iframe id="WebserverIFrame"`
	`  style="display:none"`
	`  name="WebserverIFrame"`
	`  src="/Portal/Portal.mwsl">`
	`  </iframe>`
	`  <div id="loginBox"></div>`
	`</div>`

3	In the <style> tag, the appropriate IDs or the class is accessed to change the style properties of the input fields and buttons.
	
	`<style>`
	`  /*S7-1500 */`
	
	`  #Login_Area_Name_InputTag{  /*Input field username*/`
	`  width:100px;`
	`  }`
	`  #Login_Area_PW_InputTag{  /*Input field password*/`
	`  width:100px;`
	`  }`
	`  #Login_Area_SubmitButton{  /*Button for login*/`
	`  width:80px;`
	`  }`
	`  .Logout_Button{  /*Button for logout*/`
	`  width: 100px;`
	`  }`
	`  #logout_user_name{  /*div of username after login*/`
	`  font-family: Arial,Helvetica Neue,Helvetica,sans-serif;`
	`  }`

© Siemens AG 2018 All rights reserved

17.2.2.2 Demo17.js structure

<table>
<tr><th>No.</th><th>Code description</th></tr>
<tr><td>1</td><td>Determining the CPU type (S7-1200 or S7-1500)
In the “Demo12.js” JavaScript file, a function to determine whether an S7-1200 or S7-1500 is used is executed in “$(document).ready(function(){}”.
<pre>
$(document).ready(function(){
    if (plcType != "1200" && plcType != "1500")
    {
        $.ajax({type:"GET",url:"../../../Portal/Intro.mwsl",data:"",dataType:"text"})
                .done(function(webpageData){
                    var search12 = webpageData.search("CPU12");
                    var search15 = webpageData.search("CPU15");
                    if (search12 >= 0)
                    {
                        plcType = "1200";
                        $.init();
                    }
                    else if (search15 >= 0)
                    {
                        plcType = "1500";
                        $.init();
</pre>
Alternatively, you can directly call the functions of “$.init” in “$(document).ready(function[]{}”.
Replace “plcType” with “1200” or “1500”, including quotation marks.</td></tr>
<tr><td>2</td><td>This function transfers the CPU type to the S7 Framework.
<pre>
$.init = function(){
    S7Framework.initialize(plcType, "");
}
</pre></td></tr>
<tr><td>3</td><td>When the page has loaded, the system checks whether the user is already logged in via the standard web page or a previous login operation.
If no user is logged in, a pop-up appears that reads:
“Please log in!”
<pre>
$(window).load(function(){
    if(S7Framework.loginCheck()){
    }
    else{
        alert("Please log in!");
    }
});
</pre>

</td></tr>
</table>

© Siemens AG 2018 All rights reserved

18 High-Performance Communication via a String

When data is exchanged between a web server and a web browser, variables are transferred individually. If you want to implement comprehensive, high-performance data exchange between the web server and the controller, this is achieved by communication of a concatenated string. As another measure to increase performance, use AJAX.

AJAX

AJAX (“Asynchronous JavaScript and XML”) provides a method for exchanging data asynchronously between a web browser and the web server. Data is exchanged without reloading the web page itself.

18.1 Automation task

High-performance transfer of larger data volumes to the controller.

Requirements for the automation task

- The values in the controller are converted to an ASCII string with a maximum of 255 characters.
- ASCII strings are cyclically read using the S7 Framework (JavaScript) and split into single variables.
- The values are individually sent to the controller.
- The values are sent from a form to the controller.

Requirement for communication between STEP 7 and the user-defined page using AJAX

Updating and writing PLC tags using AJAX is implemented on the user-defined page. To this end, the “json” file format must be added to the user-defined page properties in “Files with dynamic content:”.

Figure 18-1: Requirement – update using AJAX with json in S7

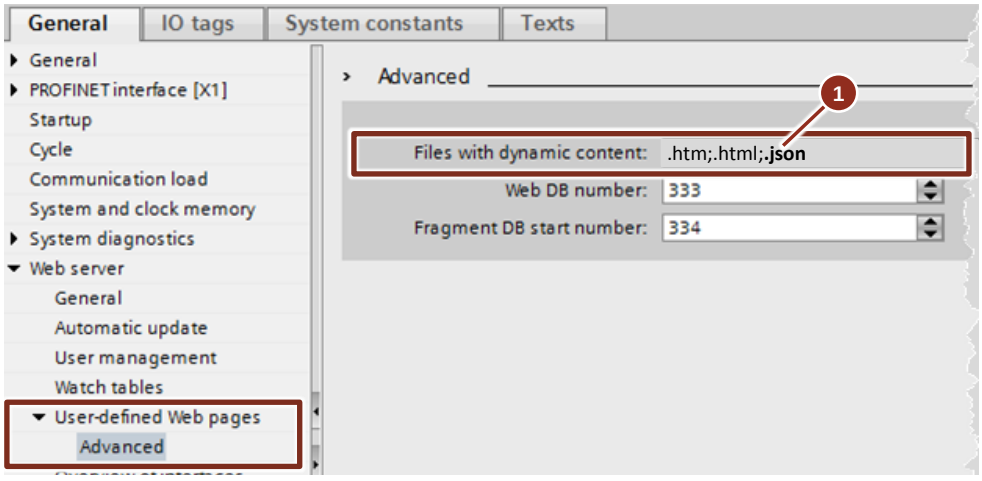


Table 18-1: Instructions for adding a file format

Step	Instruction
1	In the input field, add the “json” file extension to “.htm;.html;”.

© Siemens AG 2018 All rights reserved

18.2 Functional mechanisms and use

The user-defined page simulates a package (gray rectangle) that moves from the left to the right in the display area bordered in black. The value entered in “Speed” controls the speed of the package.

When the package reaches the sensor (green circle), this sensor changes from “green” to “red”. A black horizontal line marks the sensor’s detection range; a count below this line shows the detected packages.

The [send] button transfers the information entered for name and age to the controller.

Figure 18-2: User-defined page for high-performance data transfer using AJAX

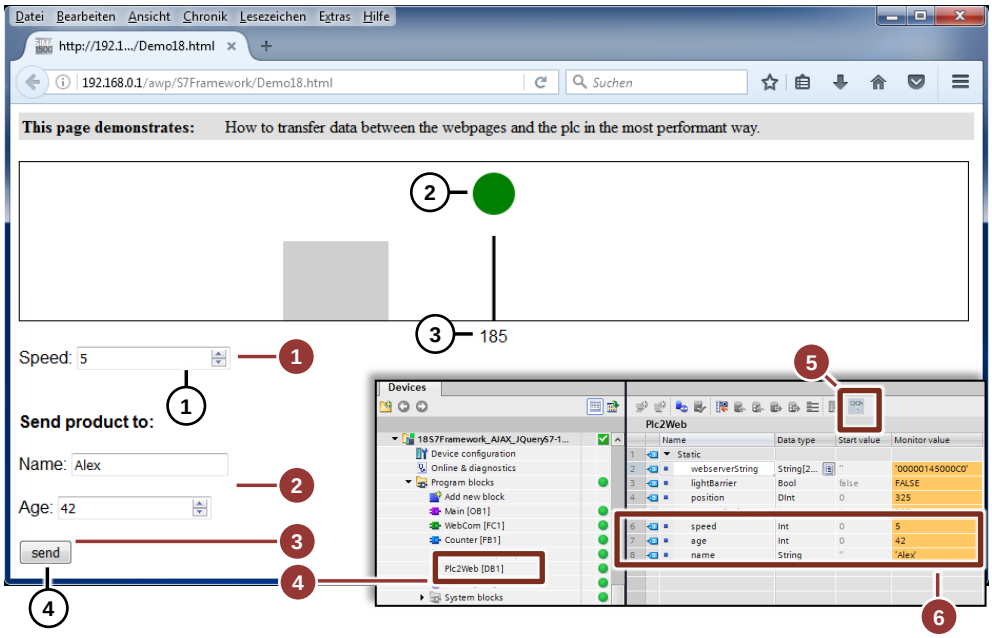


Table 18-2: User-defined page for high-performance data transfer using AJAX

Item	Description
1	With the “Speed” value, you influence the speed of the package.
2	Signal state of the sensor: (green = free; red = package detected)
3	Count that shows the packages that have already passed.
4	[send] transfers the “name” and “age” to the controller.

Table 18-3: Instructions for using the user-defined page

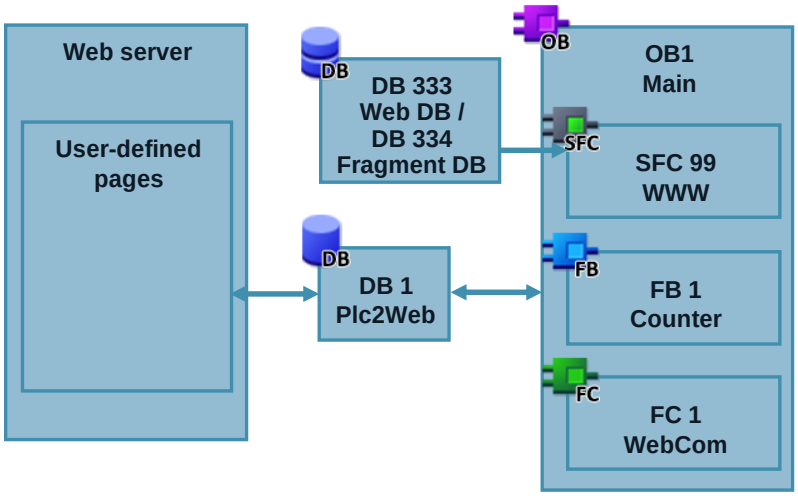
Step	Instruction
1	Change the “Speed”.
2	Enter the “name” and “age” of the package’s recipient.
3	Select the “send” button.
4	In TIA Portal, open the “Plc2Web” data block.
5	Click “Monitor”.
6	Verify that the name and age have been received.

© Siemens AG 2018 All rights reserved

18.2.1 S7 program structure

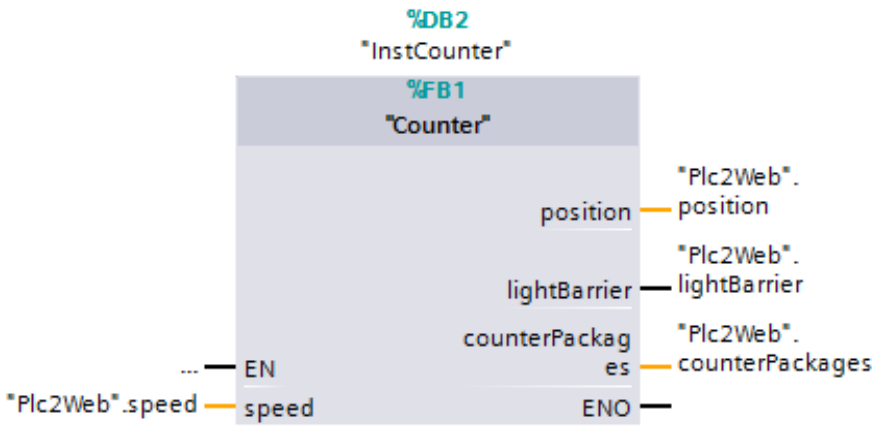
The WWW function from OB 1 is called in the S7 program.
FB 1 “Counter” is used for simulating the package and the sensor.
FC 1 “WebCom” is used for converting the variables to a string.

Figure 18-3: S7 program for high-performance data transfer using AJAX

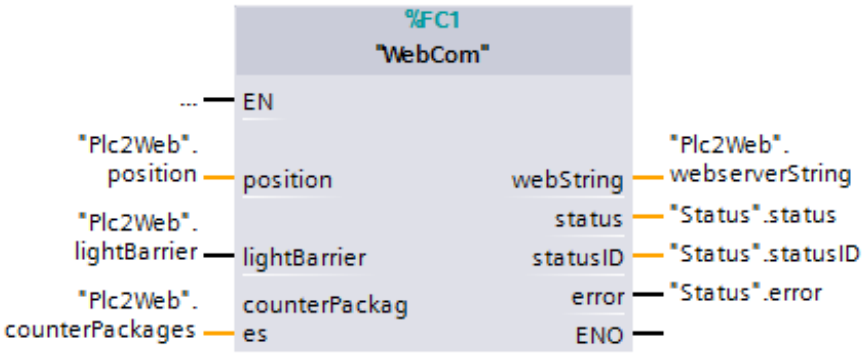


© Siemens AG 2018 All rights reserved

How the PLC program works

No.	Function
1	The first block call in OB 1 is the WWW function (SFC 99). For more information about the function, please refer to Chapter 4.2.
2	**FB 1 “Counter” – (simulation)** The function block calculates the following output variables as simulation values for the sequence displayed on the web page. position: calculates the package’s position. lightBarrier: checks whether the package is below the sensor. counterPackages: If the sensor detects a package, a count is incremented. 

© Siemens AG 2018 All rights reserved

<table>
<tr><th>No.</th><th>Function</th></tr>
<tr><td>3</td><td>
FC 1 “WebCom” – (web communication)
The function initially converts the simulation values of FB 1 to a “webString” variable. The “webserverString” string variable follows from this variable.


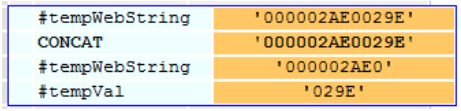
If you want to transfer additional values, you must extend the string in this block.

The TIA Portal system function “HTA” converts the following variable types to an ASCII string.

SINT
INT
LINT
DINT

Example:
<pre>// Convert DINT to ASCII-String
#tempRetValHTA := HTA(IN := #position, N := 4, OUT => #tempWebString);</pre>
If you want to convert variables of the REAL or LREAL type, make sure to first convert them to DWORD or LWORD. Then convert these variables to DINT or LINT.

A bit is converted to an ASCII string using an IF statement.
<pre>General:

IF bool THEN
   char := '1';
ELSE
   char := '0';
END_IF;</pre>
Example:
<pre>// Convert BOOL to ASCII-String
IF #lightBarrier THEN
  #tempVal := #TRUE;
ELSE
  #tempVal := #FALSE;
END_IF;</pre>
The “CONCAT” function concatenates two ASCII strings.
<pre>// Add variable 1 to variable 2
#tempWebString := CONCAT(IN1 := #tempWebString, IN2 := #tempVal);</pre>
Example:


</td></tr>
</table>

© Siemens AG 2018 All rights reserved

No.	Function
4	**DB 1 "Plc2Web"** DB1 contains all variables that are sent from the CPU to the user-defined page or read. The data block also contains the concatenated string variable, "webserverString". **Plc2Web** Name / Data type / Start value / Monitor value 1 Static 2 webserverString / String[254] / '' / '000002AE0029E' 3 lightBarrier / Bool / false / FALSE 4 position / DInt / 0 / 686 5 counterPackages / Int / 0 / 670 6 speed / Int / 0 / 0 7 age / Int / 0 / 42 8 name / String / '' / 'Alex'

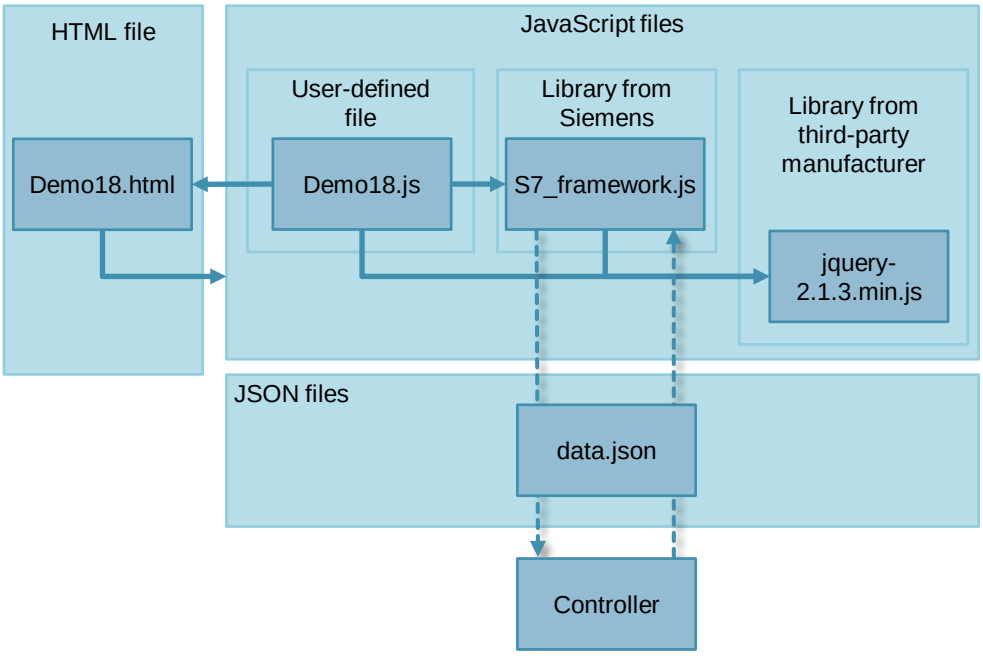
18.2.2 User-defined web page structure

The diagrammatic overview below shows the basic file structure of the user-defined page.

This description focuses on the JavaScript file's code elements, which makes basic knowledge of HTML necessary for this application example.

User-defined page structure

Figure 18-4: Diagrammatic overview of the user-defined page



© Siemens AG 2018 All rights reserved

Demo18.html

All JavaScript files are integrated into the “Demo18.html” file.

Demo18.js

The “Demo18.js” JavaScript file controls read/write from/to the controller by calling functions from the “S7_framework.js” library.

S7_framework.js

The JavaScript file contains a number of functions for converting and processing data from a SIMATIC S7.
For more information about the S7 Framework, please refer to Chapter 4.3.

The Demo18.js file uses S7 Framework functions to transfer data via the JSON files.

data.json (JSON)

The file is used for reading/writing the concatenated string using the functions within the S7 Framework.

jquery-2.1.3.min.js (jQuery library)

The jQuery library file allows more efficient programming with JavaScript. For more information about the jQuery library, please refer to Chapter 4.3.

© Siemens AG 2018 All rights reserved

18.2.2.1 Demo18.html structure

No.	Code description

1	The JavaScript files used are integrated into the HTML document's <head> with the below syntax:
	
	`<head>`
	`  <meta http-equiv="content-type" content="text/html; charset=UTF-8">`
	`  <script type="text/javascript" src="script/jquery-2.1.3.min.js"></script>`
	`  <script type="text/javascript" src="script/S7_framework_0.1.7.js"></script>`
	`  <script type="text/javascript" src="script/Demo18.js"></script>`
	`  <script type="text/javascript" type="text/javascript"></script>`
	`</head>`
	The "src" property specifies the respective file path.

18.2.2.2 Demo18.js structure

No.	Code description

1	**Determining the CPU type (S7-1200 or S7-1500)**
	In the "Demo18.js" JavaScript file, a function to determine whether an S7-1200 or S7-1500 is used is executed in "$(document).ready(function(){}".
	`$(document).ready(function(){`
	`  if (plcType != "1200" && plcType != "1500")`
	`  {`
	`  $.ajax({type:"GET",url:"../../../Portal/Intro.mwsl",data:"",dataType:"text"})`
	`  .done(function(webpageData){`
	`  var search12 = webpageData.search("CPU12");`
	`  var search15 = webpageData.search("CPU15");`
	`  if (search12 >= 0)`
	`  {`
	`  plcType = "1200";`
	`  $.init();`
	`  }`
	`  else if (search15 >= 0)`
	`  {`
	`  plcType = "1500";`
	`  $.init();`
	
	Alternatively, you can directly call the functions of "$.init" in "$(document).ready(function[]{}".
	Replace "plcType" with "1200" or "1500", including quotation marks.

2	This function transfers the CPU type to the S7 Framework.
	
	`$.init = function(){`
	`  S7Framework.initialize(plcType, "");`

3	The "S7Framework.readData" function reads the PLC tags specified in the selected json file (data.json). When the function has been executed, the specified event handler (updateValues) is run.
	
	`S7Framework.readData("script/data.json",...`
	`  "init read data",updateValues);`
	...

4	The "updateValues" function further processes the read values. The values are transferred to the "values" array. First value = values[0] Second value = values[1] … etc. `function updateValues(values)`

© Siemens AG 2018 All rights reserved

18.2.2.3 data.json structure

No.	Code description

1	**data.json**
	The “data.json” file contains the information for reading the string as well the parameters of the string and the UDP commands of the PLC tags to be written.
	
	The information for writing the string corresponds to the information for reading.
	
	The string to be read is defined: Plc2Web.webserverString
	The string structure is defined: Length; type; other string variables
	
	`<!-- AWP_In_Variable Name='"Plc2Web".speed' -->`
	`<!-- AWP_In_Variable Name='"Plc2Web".name' -->`
	`<!-- AWP_In_Variable Name='"Plc2Web".age' -->`
	`{`
	`"val" : ":="Plc2Web".webserverString:",`
	`"len" : "8;1;4",`
	`"typ" : "2;0;2",`
	`"str" : ""`
	`}`
	
	DINT (DINT) + BOOL (BOOL) + INT (INT)
	
	“val” corresponds to the string to be read.
	
	“len” describes the “length” of the variable (variable type)
	1 = 1 bit => corresponds to: BOOL
	2 = 8 bits => corresponds to: BYTE
	4 = 16 bits => corresponds to: INT / WORD
	8 = 32 bits => corresponds to: DINT / DWORD / REAL
	16 = 64 bits => corresponds to: LINT / LWORD / LREAL
	
	“typ” describes the variable type (defined in the S7 Framework)
	0 = BOOL;
	1 = UINT;
	2 = INT/DINT;
	3 = REAL;
	4 = LREAL;
	5 = STRING;
	
	“str” is used for loading variables that were not converted to the string.

© Siemens AG 2018 All rights reserved

19 Installation

19.1 Installing the hardware and software

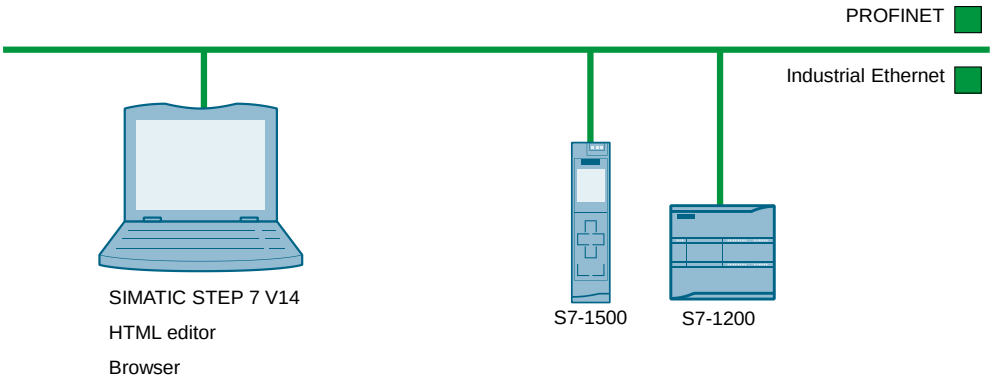
Installing the hardware

The following figure shows the hardware configuration for the examples.

The PC (web browser) is connected to the CPU (web server) via Industrial Ethernet using the PN interface.

The SIMATIC S7 controllers were used separately as the examples use only one IP address for the S7 controller.

Figure 19-1: Hardware configuration of the examples



Note Follow the installation and connection guidelines provided in the appropriate manuals and installation instructions.

Installing the software

Table 19-1: Installing the software packages for the examples

No.	Action
1.	Install SIMATIC STEP 7 Professional (TIA Portal).
2.	Install a tool for creating the user-defined page, e.g. MS Expressions, Notepad++, on the PC that you want to use to create the user-defined page.
3.	Install a web browser, e.g. Internet Explorer or Firefox, on the PC that you want to use to access the CPU's web pages.

© Siemens AG 2018 All rights reserved

19.2 Installing the application example

Table 19-2: Procedure for using the examples

No.	Action	Comment
1.	Unzip the “68011496_examples_for_S7WebServer_CODE_v11 .zip” file to your project directory.	-
2.	Start SIMATIC STEP 7 V14.	-
3.	In SIMATIC STEP 7 V14, open the project.	-
4.	Select the desired program. The programs are numbered by the chapter numbers in the documentation.	-
5.	Go to the Device view.	-
6.	If you want to use a different CPU, replace the CPU in the hardware catalog.	-
7.	In the CPU properties of the Ethernet interface, assign the IP address of your CPU.	-
8.	Select the CPU and download the entire project to the CPU.	-
9.	Start a web browser and use the IP address to open the web page of your CPU.	For more information, please refer to the “Use” chapter of the examples.

© Siemens AG 2018 All rights reserved

20 Internet links

This table provides a selection of links for related information.

Table 20-1: Internet links

No.	Topic
/1/	Siemens Industry Online Support http://support.automation.siemens.com
/2/	Reference to the entry https://support.industry.siemens.com/cs/ww/en/view/68011496
/3/	HTML, CSS, JavaScript, http://www.w3schools.com
/4/	S7-1500 Web Server Function Manual https://support.industry.siemens.com/cs/ww/en/view/59193560
/5/	SVG, w3schools http://www.w3schools.com/svg/default.asp
/6/	jQuery Flot http://www.flotcharts.org/
/7/	jQuery https://jquery.com/

© Siemens AG 2018 All rights reserved

21 History

Table 21-1: Document history

Version	Date	Modifications
V1.0	08/2015	First version
V 2.0	04/2017	The following examples were revised: • Change language • Creating a graph The following examples were added: • Creating display elements • Changing pictures using JavaScript • Integrating a button for restarting the CPU • Integrating the login option • High-performance data transfer using variables converted to strings
V 2.1	08/2018	Example 18: Code adaptions

© Siemens AG 2018 All rights reserved